Topological Constraints and Ordering in Model Frustrated Magnets

— — —

Contraintes Topologiques et Ordre dans les Systèmes Modèle pour le Magnétisme Frustré

Adam Harman-Clarke

University College London

and

École Normale Supérieure de Lyon

Supervisors:

Professor Steven T. Bramwell

and

Professor Peter C. W. Holdsworth

This thesis is submitted toward the degree of

Doctor of Philosophy

This thesis is dedicated to my Grandad.

He started walking down the path that led me here.

Declaration of authenticity

I, Adam Harman-Clarke, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signed: .....

Note

This thesis is presented as an electronic pdf document and is designed to be read in that form. All references and citations are shown in blue and are clickable links. The results section contains animations of the neutron scattering behaviour which only function when the document is viewed using Adobe Reader. The paper version of this thesis contains representative images from the animation in its place which portray the same information. The complete code of the programs written during this work is included as appendices in the electronic document only as this information is considerably more useful if it can be copied into a text editor rather than read.

Acknowledgements

My first thanks and gratitude goes to my supervisors, Professor Steve Bramwell at UCL and Professor Peter Holdsworth at ENS Lyon. It would, of course, have been impossible to carry out this work without their academic guidance and considerable wisdom; however, anybody who has undertaken a doctorate will be aware of the academic contribution of their supervisors. I would particularly like to thank Peter and Steve for inspiring me to be excited and interested in this project and remaining so themselves, especially during the difficult periods when it seemed we were clinging onto the promise of, rather than striding toward successful results. Thanks is also due for their patience when I did not understand something and their calm and relaxed manner which makes them great teachers as well as scientists. Their individual inputs are complimentary in a way that reflects years of combined effort and I feel very lucky to have had the chance to work together with them.

I have received support and guidance from many people during my doctorate but I would particularly like to mention Björgvin Hjörvarsson who presented me with the opportunity for an interesting collaboration which has become a valuable part of my thesis but also took a real interest in my work and the overall progression of my thesis. My visit to the Ångströmlaboratoriet to work with Vassilis, Unnar and the rest of his group is a great memory from the last four years. I would also like to thank Andrea Taroni who helped me from the very beginning of my project and has been supportive and encouraging ever since, Simon Banks who has also been exceptionally generous with his time and provided me with some really useful code and Ludovic Jaubert who helped me a lot whilst I was visiting the ENS.

Thanks also to Professor Catlow, then head of department of UCL Chemistry, who helped to find the funding that allowed me to start my doctorate and the EPSRC for providing it, to Professor Hjörvarsson who made a big contribution during my final year once my initial funding had run out and particularly to Peter and ENS Lyon who have given me a great deal of financial support throughout my studies and especially during the last year.

Moving outside of the academic sphere, I would like to thank my extended family.

I am very grateful to have such a supportive and large group of people around me who have all contributed toward my education in many ways. Having the opportunity to be a student to a doctoral level is a great privilege that I have only been able to enjoy due to their personal support and financial generosity and I will never forget that.

A special mention goes to Doug, Dave and Dara for the crosswords, the conversations and the beer; all welcome distractions from work!

Finally I would like to thank Daphne who has put up with a lot, particularly in the last six months, whilst I have worked on my doctorate and who has unswervingly supported me, encouraged me, distracted me, reproached me and energised me, all in exactly the right measure!

Abstract

In this thesis a series of model frustrated magnets have been investigated. Their common parent is the spin ice model, which is transformed into the kagome ice and square ice models in two-dimensions, and an Ising spin chain model in one-dimension. These models have been examined with particular interest in the spin ordering transitions induced by constraints on the system: a topological constraint leads, under appropriate conditions, to the Kasteleyn transition in kagome ice and a lattice freezing transition is observed in square ice which is due to a ferromagnetic ordering transition in an Ising chain induced solely by finite size effects.

In all cases detailed Monte Carlo computational simulations have been carried out and compared with theoretical expressions to determine the characteristics of these transitions. In order to correctly simulate the kagome ice model a loop update algorithm has been developed which is compatible with the topological constraints in the system and permits the simulation to remain strictly on the groundstate manifold within the appropriate topological sector of the phase space.

A thorough survey of the thermodynamic and neutron scattering response of the kagome ice model influenced by an arbitrary in-plane field has led to a deeper understanding of the Kasteleyn transition, and a computational model that can predict neutron scattering patterns for kagome ice materials under any experimental conditions. This model has also been shown to exhibit quantised thermodynamic properties under appropriate conditions and should provide a fertile testing ground for future work on the consequences of topological constraints and topological phase transitions. A combined investigation into the square ice and Ising chain models has revealed ordering behaviour within the lattice that may be decoupled from underlying ferromagnetic ordering and is particularly relevant to magnetic nanoarrays.

keywords: topological constraint, spin ice, kagome ice, square ice, ising chain, frustration, spin model, Kasteleyn, phase transition, finite size scaling, Monte Carlo, cluster algorithm, neutron diffraction, computer simulation.

Résumé

Dans cette thèse, l'étude de plusieurs modèles de systèmes magnétiques frustrés a été couverte. Leur racine commune est le modèle de la glace de spin, qui se transforme en modèle de la glace sur réseau kagome (kagome ice) et réseau en damier (square ice) à deux dimensions, et la chaîne d'Ising à une dimension. Ces modèles ont été particulièrement étudiés dans le contexte de transitions de phases avec un ordre magnétique induit par les contraintes du système: en effet, selon la perturbation envisagée, les contraintes topologiques sous-jacentes peuvent provoquer une transition de Kasteleyn dans le kagome ice, ou une transition de type vitreuse dans la square ice, due à l'émergence d'un ordre ferromagnétique dans une chaîne d'Ising induit seulement par des effets de taille fini.

Dans tous les cas, une étude détaillée par simulations numériques de type Monte Carlo ont été comparées à des résultats théoriques pour déterminer les propriétés de ces transitions. Les contraintes topologiques du kagome ice ont requis le développement d'un algorithme de vers permettant aux simulations de ne pas quitter l'ensemble des états fondamentaux.

Une revue poussée de la thermodynamique et de la réponse de la diffraction de neutrons sur kagome ice sous un champ magnétique planaire arbitraire, nous ont amené à une compréhension plus profonde de la transition de Kasteleyn, et à un modèle numérique capable de prédire les figures de diffraction de neutrons de matériau de kagome ice dans n'importe quelles conditions expérimentales. Sous certaines conditions, ce modèle a révélé des propriétés thermodynamiques quantifiées et devrait fournir un terreau fertile pour de futurs travaux sur les conséquences des contraintes et transitions de phases topologiques. Une étude combinée du square ice et de la chaîne d'Ising a mise en lumière l'apparition d'un ordre sur réseau potentiellement découplé de l'ordre ferromagnétique sous-jacent, et particuliérement pertinent pour les réseaux magnétiques artificiels obtenus par lithographie.

mots clés: contraintes topologiques, glace de spin, glace de kagome, glace sur réseau en damier, chaîne d'Ising, frustration, modèles de spin, Kasteleyn, transitions de phases, effets de taille finie, Monte Carlo, algorithme de cluster, diffraction de neutron, simulations numérique.

Contents

List of Figures

Symbols and Abbreviations

Vector quantities are in bold font.

Symbols	
$\langle \dots \rangle$	Indicates a thermal average
$\lvert \dots \rvert$	Indicates the absolute value
$\mathcal{A}$	General quantity
a_{ij}	Transition rate between states i and j
b_r	Scattering length
C	Specific heat
$\mathbf{d}$	Distance between scattering planes
E	Energy
E_{Z}	Zeeman energy
e	Electronic charge
$\mathcal{F}$	Helmholtz energy
$\mathcal{G}$	Gibbs energy
g_J	Landé g-factor
g_S	Spin g-factor
g	Degeneracy
H	Applied magnetic field
$\mathcal{H}$	Hamiltonian
h	Planck's constant
I	Electrical current
J	Nearest neighbour coupling interaction
$\mathcal{K}$	Kinetic energy
$\mathbf{k}$	Wavevector
k_B	Boltzmann's constant
L	Orbital angular momentum
M	Magnetisation

Continued on next page

– continued from previous page

Symbols	
m	Mass
N	Number of objects
$\mathbf{P}$	Boltzmann distribution
P	Probability
$\mathbf{p}$	Momentum
$\mathbf{Q}$	Neutron scattering vector
$\mathbf{R}$	Reciprocal lattice vector
R	Shape factor
$\mathbf{r}$	Position *or* lattice vector
r_0	Magnetic scattering length
r	A general distance
$\mathbf{S}$	Spin angular momentum
t	Time
T	Temperature
T_C	Critical temperature
T_K	Kasteleyn critical temperature
$\mathcal{U}$	Potential energy
v	Linear velocity
w	Boltzmann weight
W	Winding number
$\mathcal{Z}$	Partition function
z	Fugacity
α	Specific heat critical exponent
β	Order parameter critical exponent *or* inverse temperature
γ	Susceptibility critical exponent
Γ	Commensurability
δ	Critical isotherm exponent

Continued on next page

– continued from previous page

Symbols	
η	Correlation function critical exponent
θ	Angle
λ	Wavelength
μ	Total magnetic moment
μ_B	Bohr magneton
μ_L	Orbital magnetic moment
μ_S	Spin magnetic moment
ν	Correlation length critical exponent
ξ	Correlation length
σ	Scattering cross section
Ω	Solid angle
τ	Coulomb interaction constant *or* reduced scaled temperature
ϕ	Magnetic field angle on the kagome plane
χ	Magnetic susceptibility

1 Introduction to Model Systems and Collective Effects

Arguably the most common experience of phase transitions is in the transitions between ice, water and steam but magnetic materials have often been used as a 'testing ground' for exploring these phenomena and the ferromagnetic phase transition is often quoted as the prototypical example of the onset of an order parameter in Landau's formulation of phase transitions [1]. In the last seventy years computational investigations have become increasingly important in research and magnetic systems are well suited to numerical modelling. Generally the individual elements of the system are well understood as are the interactions between them and these can be easily modelled using Monte Carlo (MC) or other numerical techniques.

Computational models are almost always simplifications of the subject they are modelling and a particular benefit of magnetic systems is that the (often quite) simple models represent their more complicated subjects very well and therefore produce results that are computationally inexpensive whilst still being accurate enough to match experimental work and make useful predictions. This thesis is based on the computational modelling of reasonably simple magnetic models which produce results that can aid in the understanding of the more complicated materials they represent.

A running theme throughout this work is the presence of phase transitions, effects due to the finite size of samples and models, and reduced dimensionality in the models; hence the rest of this chapter is an introduction to magnetism and frustration, phase transitions and lattice models and is completed by a summary of the aims of the work presented in this thesis. The following two chapters complete the necessary background for the thesis by examining real systems that are experimental realisations of the lattice models, and then motivating numerical modelling and providing detail of the techniques used for both general and more specific simulations. Chapters (4), (5) and (6) are dedicated to kagome ice, a model which is the focus of a large proportion of this thesis. First an account of the Kasteleyn transition

is presented followed by the more interesting aspects of the kagome ice model and the specialist simulation techniques necessary to study it and finally a review of the simulation results including some comparison with experiment is provided. Chapter (7) discusses the square ice model and presents the results of simulations performed on this system. A comparison is then drawn between these results and experimental observations of an artificial spin ice nanoarray. Conclusions are presented in chapter (8) in the form of a review looking backward over the time spent carrying out this work and finally the appendices contain supplementary material including, in the electronic document only, the complete code of all the simulation models.

1.1 Magnetism

Here a brief semi-classical derivation of how the angular momentum of an electron creates a magnetic moment is presented based on the more detailed explanation in, for example, references [2–4]. Magnetism is intrinsically associated with direction and a general description of magnetism contains many vector quantities. For simplicity, in this work the magnetic moment is defined parallel to the magnetic field unless specified otherwise hence many quantities are presented as scalars representing the vector magnitude only.

An electron of electrical charge $-e$ and mass m_e moving in a circular orbit around an isolated nucleus, assumed to be stationary, at a distance r with a linear velocity v has an orbital angular momentum,

$$L = m_e v r \tag{1}$$

The motion of charge is of course a current I and a circular current will generate a magnetic moment μ_L,

$$\mu_L = I\pi r^2 \tag{2}$$

$$= -\frac{ev}{2\pi r}\pi r^2 \tag{3}$$

$$= -\frac{e}{2m_e}L \tag{4}$$

In quantum mechanical terms observables such as magnetic moment and orbital angular momentum are represented by their corresponding operators (denoted by a $\hat{}$ above the symbol) [5]. The angular momentum operator is defined,

$$\hat{L} = -i\hbar r \times \nabla \tag{5}$$

which has eigenvalues

$$\langle \hat{L}^2 \rangle = \hbar^2 L(L+1), \quad 0 \leq L \leq L_{max}, \quad L \in \mathbb{Z}^+ \tag{6}$$

Forming the corresponding operator representation of equation (4),

$$\hat{\mu}_L^2 = \frac{e^2}{4m_e^2}\hat{L}^2 \tag{7}$$

$$\implies \langle \hat{\mu}_L^2 \rangle = \frac{e^2\hbar^2}{4m_e^2} L(L+1) \tag{8}$$

$$= \mu_B^2 L(L+1) \tag{9}$$

where,

$$\mu_B = \frac{e\hbar}{2m_e} = 9.2732 \times 10^{-24} \text{ Am}^2 \tag{10}$$

is defined as the Bohr magneton. When considering the angular momentum of a free ion it is useful to work in spherical polar coordinates and define the polar axis to be z. The operator for the component of the angular momentum on this axis is,

$$\hat{L}_z = -i\hbar \frac{\partial}{\partial \phi} \tag{11}$$

with eigenvalues

$$\langle \hat{L}_z \rangle = \hbar m_L, \quad m_L = -L, -L+1 \ldots L-1, L \tag{12}$$

The expectation of the z-component magnetic moment operator is

$$\langle \hat{\mu}_z \rangle = -\mu_B m_L \tag{13}$$

Alongside orbital angular momentum electrons also possess an intrinsic spin angular momentum which must be taken into account. Similarly to equation (4), this gives rise to a spin moment of,

$$\mu_S = -\frac{g_S \mu_B}{\hbar} S \tag{14}$$

where it is necessary to include $g_S = 2.0023$, called the g-factor for the spin. The Dirac equation [6] predicts the value $g_S = 2$, but a complete treatment yields the slightly larger value due to a correction term arising from quantum electrodynamics.

The total magnetic moment operator of an atom, $\hat{\mu}$, is due to both the spin and orbital angular momentum. The two momenta can be combined according to Hund's rules [2] when spin orbit coupling is dominant, which is the case for the rare-earth atoms considered in this work, such that the quantum number, $J = L + S$, is well-defined and,

$$\hat{\mu} = -\frac{\mu_B}{\hbar} g_J \hat{J} \tag{15}$$

where the Landé g-factor,

$$g_J = \frac{3J(J+1) + S(S+1) - L(L+1)}{2J(J+1)} \tag{16}$$

and the eigenvalues are

$$\langle \hat{\mu}^2 \rangle = \mu_B^2 J(J+1) \tag{17}$$

$$\langle \hat{\mu}_z \rangle = \mu_B m_J, \quad m_J = -J, \ldots, J \tag{18}$$

When a free ion is placed into a crystal the surrounding ions exert a non-uniform crystalline electric field onto it which perturbs the manifold of m_J states. The result

of this is that, whilst they are degenerate in a free ion, in a crystal the quantum states are split according to the point symmetry of the crystal and often this creates a ground state doublet. In the case of $Ho_2Ti_2O_7$ for example (which is a spin ice material that will be introduced later) this splitting creates a doublet ground state with a gap of about 300 K to the next state. At low temperatures this ground state effectively permits a single spin and hence when discussing magnetic materials the magnetic moment on the individual ions is often referred to as a spin.

If a material contains such spins but there are no interactions between them then it is known as paramagnetic and in the absence of an external field it will not exhibit a magnetic moment as the vector average of the moments of all the atoms in the sample will tend to zero. An external field will influence the spins to align with it and the material will behave magnetically but when the field is removed the behaviour will also cease. Interactions between spins leads to three further classifications based on the lowest energy ordering of the moments; a ferromagnetic state when the moments align parallel to each other and the material exhibits an overall magnetic moment, an antiferromagnetic state where the alignment is antiparallel and the magnetic moments cancel each other out, and a ferrimagnetic state where the moments align antiparallel but due to differences between the magnitude of the moments on different sublattices the overall effect is to produce a small magnetic moment.

1.2 Frustration

Parallel, unit length spins interacting antiferromagnetically via an exchange force of magnitude J can be represented by the Hamiltonian,

$$\mathcal{H} = J \sum_{i \neq j} \mathbf{S}_i \cdot \mathbf{S}_j \tag{19}$$

such that the most preferable spin orientation for a pair is for both to point in opposite directions. In a geometrically frustrated system the lattice geometry prevents the spins from simultaneously reaching their individual minimum energy configuration, see figure (1). In this example it is possible to align any pair of spins on

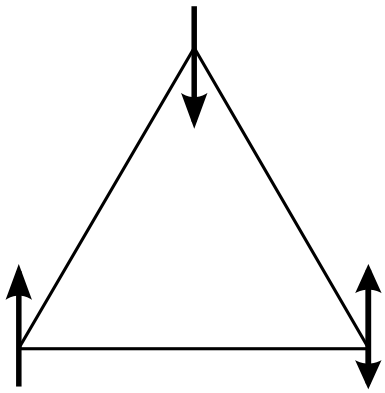

Figure 1: Spins arranged on the vertices of a triangle cannot satisfy antiferromagnetic nearest neighbour interactions. One spin, in this case the lower right corner, is always unfavourably aligned with respect to one of the other two irrespective of its orientation.

the triangle so that they satisfy antiferromagnetic interactions with respect to each other but this will always cause the remaining spin to be ferromagnetically aligned with respect to one of the other two. When extended to a triangle-based lattice this creates a higher energy state than the sum of the minimum energy between each pair of spins which is often highly degenerate and is the minimum *achievable* ground state for the system.

1.3 Phase Transitions

Empirically we observe that the state of a system is dictated by its external intensive thermodynamic parameters such as the temperature, pressure or magnetic field; if these are altered and the conjugate extensive parameters such as entropy, volume and magnetisation are free to evolve then the system can undergo a transition between adjacent phases. This suggests a useful starting point from which to consider the behaviour of phases, the equation of state. This equation relates the thermodynamic parameters for a system such that each solution is an equilibrium state of the system lying on a surface within the 'phase space'. Generally the surface described by the equation of state is projected onto two of the parameter axes at a time in order to simplify the understanding of the behaviour which produces a 'phase diagram' such as the general example shown in figure (2). The equation of state is specific to each substance and hence a phase diagram is also unique to a substance, however there are some features which are common to all phase transitions and it is these

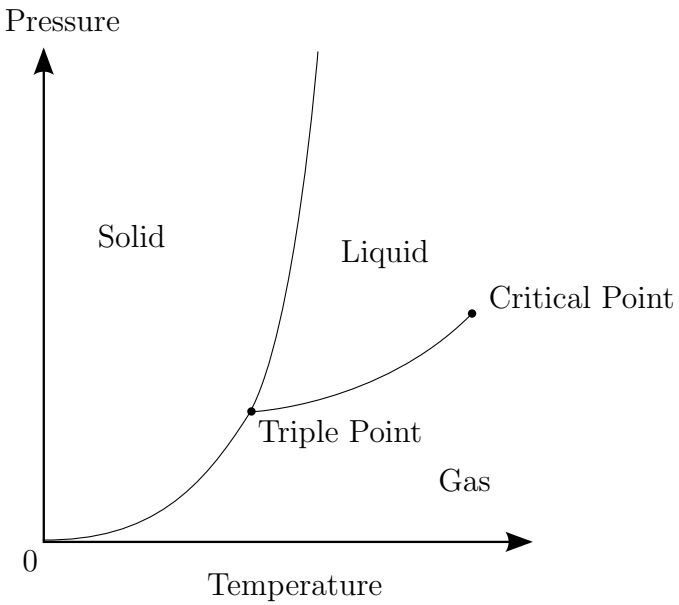

Figure 2: A general phase diagram showing the solid, liquid, and gas phases of a material. The lines between the phases are coexistence lines where the material can be found in either phase, at these points the behaviour of the free energy is non-analytic and upon passing across them the system will undergo a phase transition. At the triple point all three phases are able to coexist, and when travelling along the liquid-gas coexistence line the critical point marks the removal of the distinction between a liquid and gas.

that allow further examination and classification of phase transitions. In general the bulk free energy is analytic within the phase space and any region where this is true is called a phase, similarly the places where the free energy is not analytic are the phase transitions.

Further consideration suggests that interactions within the system must conspire with the thermodynamic parameters to determine the nature of the phase. For example the simplest model of a material, an ideal gas of non-interacting particles, does not allow phase transitions between gas, liquid or solid regardless of the temperature or pressure, each particle must interact with the others in order to form a liquid or solid phase. The different phases of a material can be classified partly by their symmetry due to these interactions.

1.3.1 Symmetry

In general the higher the energy of a material the higher the symmetry it possesses, and if this energy is decreased and the material undergoes a phase transition the symmetry will often be reduced. Generally the Hamiltonian contains the highest

symmetry possible for that system but the low energy states will only contain a reduced symmetry, for example the high temperature, paramagnetic phase of a ferromagnet has rotational symmetry whereas below the Curie temperature a direction (of the magnetisation) appears in the material and the rotational symmetry is broken. A phase transition is associated with a change in the order of a system and we can define a quantity called the 'order parameter' that has a finite value in the more ordered state and has a value of zero at the transition. If there is some symmetry broken at the phase transition then the order parameter will be related to that symmetry. In the case of the ferromagnet the order parameter is the spontaneous magnetisation which is a vector quantity and vectors are defined with a direction which breaks the rotationally invariant space they are described in just as the onset of magnetisation in a ferromagnet breaks the rotationally invariant symmetry of the paramagnetic phase.

1.3.2 First-Order and Continuous Transitions

The behaviour of the order parameter around a phase transition provides another method of classification for phase transitions. Often the order parameter is a first derivative of the energy of the system with respect to a suitable intensive thermodynamic parameter. For a magnetic system described by an energy $\mathcal{U}(S, V, M, N_1 \dots N_r)$ these are the magnetisation, M, and the magnetic field, H, respectively

$$M = \left(\frac{\partial \mathcal{U}}{\partial H}\right)_{S,V,N_1\dots} \tag{20}$$

If this first derivative is non-analytic across the transition then it is referred to as a first-order transition, and if it is continuous but a higher order derivative of the energy is discontinuous (or infinite) the transition is called continuous. This classification, due to Fisher, is more appropriate than the Ehrenfest classification which was derived before a detailed understanding of the non-analytic behaviour near to a transition showed that the second (or higher) derivative can be divergent or discontinuous [3]. Typically the order parameter is zero above the transition and can be determined below the transition by minimising the energy with respect to

one of the external parameters, as illustrated for a ferromagnetic transition above. Order parameters of this type are known as non-conserved, however it is possible for the order parameter to be conserved. In the case of the liquid-gas transition the order parameter is the difference between the density of the phase and its value at the critical point and above the transition the density of the system is still a well defined quantity determined by the external constraints on the system [7].

1.3.3 Mean field and Landau theory

The quantum mechanical calculation of the magnetisation of an ideal paramagnet shows that it has the form [2],

$$M = N g_J \mu_B J \mathcal{B}_J(x) \tag{21}$$

where

$$\begin{aligned} N &= \text{Number of spins} \\ g_J &= \text{Landé } g\text{-factor} \\ J &= \text{Total angular momentum quantum number} \\ \mathcal{B}_J(x) &= \text{Brillouin function} \\ x &= \frac{g_J \mu_B J H}{k_B T} \end{aligned}$$

and the Brillouin function is,

$$\mathcal{B}_J(x) = \frac{2J+1}{2J} \coth\left(\frac{2J+1}{2J} x\right) - \frac{1}{2J} \coth\left(\frac{x}{2J}\right) \tag{22}$$

In a paramagnetic phase the argument x will be small as either the temperature is large or the field is small hence we may make the approximation,

$$\coth(x) \approx \frac{1}{x} + \frac{x}{3} + \ldots \quad x \ll 1 \tag{23}$$

Substituting equation (23) into equation (22) and then into equation (21) gives an approximation for the susceptibility known as the Curie law,

$$\chi = \frac{M}{H} \approx \frac{NJ(J+1)g_J^2\mu_B^2}{3k_BT} = \frac{C}{T} \tag{24}$$

where the terms have been gathered into the Curie constant, C, which contains the effective number of Bohr magnetons per spin, $\mu = g\sqrt{J(J+1)}\ \mu_B$.

We now consider the addition of interactions between the paramagnetic components as we have seen that these are necessary for the existence of phases and phase transitions. One of the simplest approaches toward modelling the consequences of interactions is using mean field theory. Weiss attempted to understand the behaviour of a ferromagnet by suggesting that each spin experiences a mean field, b, due to the presence of the other spins in the system which governs its average magnetic moment, and, as all spins are equal, this must be the average magnetic moment of the system [8]. The important assumption in this theory is that the spins behave in such a way that they can be represented as an average quantity, which is equivalent to stating that fluctuations around the average value of the magnetic moment are not important. As the mean field is due to the effect of all the other spins in the system it must be proportional to the total magnetisation of the system

$$b = \lambda M \tag{25}$$

where

$$\lambda M = \sum_{j \neq i} J_{ij} < |S|_j > \tag{26}$$

indicates the average moment of all other spins in the system combined with the interactions between them. Although the mean field is presented as a magnetic field it is an exchange field and typically has a much larger magnitude than the real magnetic field within the material. In Weiss' derivation the mean field is treated as an additional magnetic field and so it is assumed to influence the magnetisation of

the system, from equation (24),

$$M = \frac{C}{T}H \tag{27}$$

becomes

$$M = \frac{C}{T}(H + b) \tag{28}$$

$$= \frac{C}{T}(H + \lambda M) \tag{29}$$

$$\implies \chi = \frac{M}{H} \approx \frac{C}{T - C\lambda} \tag{30}$$

This is perhaps the simplest model to exhibit a phase transition. There is a particular temperature, $T_C = C\lambda$, called the Curie temperature at which the susceptibility has a singularity and so the magnetisation may be finite even if the field is zero and so the system spontaneously develops a magnetic moment. In fact this is true at all temperatures below the Curie temperature and this magnetisation defines a positive or negative direction which reduces the symmetry of the high temperature phase as predicted. A success of the Weiss mean field model is that the magnetic susceptibility above T_C exhibits a modified form of the Curie law, equation (24), known as the Curie-Weiss law, equation (30), specifically that the inverse susceptibility is linear as a function of temperature and goes to zero at $T = T_C$ [9].

Returning to the initial expression for the magnetisation, equation (21), it is possible to examine the behaviour of the magnetisation near to the transition. For low temperatures

$$\mathcal{B}_J(x) \to 1 \quad x \gg 1 \tag{31}$$

hence

$$M(T = 0) = M_0 = N g_J \mu_B J \tag{32}$$

and below the Curie temperature the magnetisation may be expressed by,

$$M = M_0 \mathcal{B}_J(x) \tag{33}$$

Considering atoms with $S = \frac{1}{2}$ the Brillouin function simplifies so that

$$M = M_0 \tanh(x'M) \tag{34}$$

where the argument of the Brillouin function has been modified by replacing the external field with the exchange field. This equation is implicit but by considering each side separately one positive and one negative solution for $T \leq T_C$ can be identified via a graphical solution. Equation (34) can be simplified by defining a reduced magnetisation $m = \frac{M}{M_0}$ and a reduced temperature $t = \frac{1}{x'M_0} = \frac{T}{T_C}$ to give,

$$m = \tanh(\frac{m}{t}) \tag{35}$$

and plotting both sides of the equation individually to examine where they intersect, see figure (3). This shows that at high temperatures there is only one intersection corresponding to a single solution with zero magnetisation. As the temperature is reduced the gradient of the line $y = \tanh(\frac{m}{t})$ increases for small values of the reduced magnetisation until at $T = T_C$ it is tangential to the line $y = m$ and for all temperatures below this there are three intersections of the lines corresponding to a stable, finite positive and negative magnetisation as well as the zero magnetisation solution.

Based on the ideas of interactions within a system that can be approximated with a mean-field approach and an analytic free energy throughout a single phase, Landau formulated a phenomenological theory [10, 11] to explain continuous phase transitions. The Weiss mean field theory above is based on the Helmholtz energy potential for the system defined as,

$$\mathcal{F} = \mathcal{U} - TS \tag{36}$$

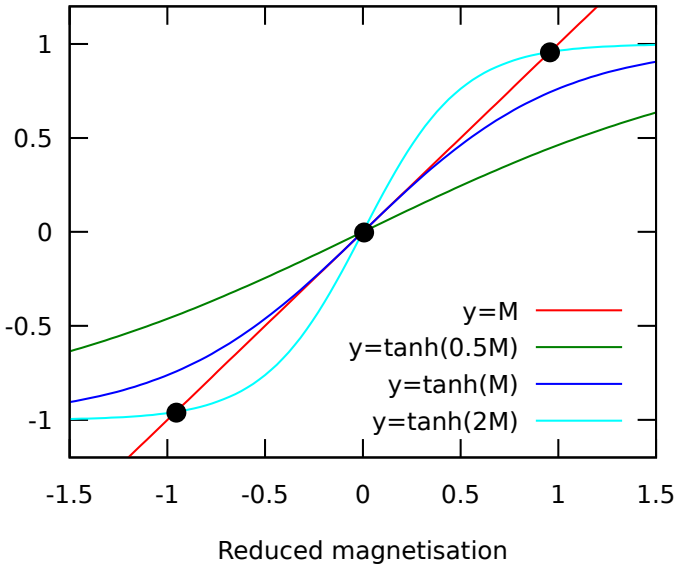

Figure 3: Plotting both sides of equation (35) for reduced temperatures $t = \frac{T_C}{2}, T_C, 2T_C$ demonstrates that for temperatures greater than or equal to T_C there are stable positive and negative solutions as well as the zero magnetisation solution which is present at all temperatures. Black dots indicate the solutions of the equation.

but Landau formulates his theory with respect to the Gibbs energy,

$$\mathcal{G} = \mathcal{F} + PV \tag{37}$$

$$= \mathcal{F} - M \cdot B \tag{38}$$

which is shown above in its standard form and in the form appropriate to magnetic systems where the pressure-volume work is generally unimportant as most magnetic systems are approximately incompressible and the relevant term is the moment-field work [1].

Given that there is an order parameter ϕ for any system, near to the transition the order parameter must be small, the temperature must be close to the critical temperature and the Gibbs potential can be represented in a Taylor series expansion of ϕ,

$$\mathcal{G} = G_0 + G_1\phi + G_2\phi^2 + G_3\phi^3 + \dots \tag{39}$$

The Gibbs potential and therefore the coefficients G_n are functions of T, P, ϕ, N_1, N_2,...,N_r. In general the expression for the Gibbs potential must obey the symmetry of the Hamiltonian, hence as the order parameter may be a vector whilst

the Gibbs potential is a scalar with an arbitrary zero value, terms in odd powers of ϕ must be removed and G_0 is irrelevant. This can be verified by the fact that odd terms would break the symmetry at all temperatures but we require it to be continuous above the critical temperature hence,

$$\mathcal{G}(T, P, \phi, N_1, N_2, \ldots, N_r) = G_2\phi^2 + G_4\phi^4 + \ldots \tag{40}$$

In the stable high temperature single phase region the Gibbs potential must be a simple minimum hence the series must have a positive coefficient to dominate the behaviour far from the transition and in this case it is sufficient to set the coefficient $G_4 > 0$. This prevents the unphysical situation where the potential could always be minimised by allowing $\phi \to \infty$. In the analysis of a first order transition it it necessary to have $G_4 < 0$ hence $G_6 > 0$ must be included in the expansion.

In order to have a simple minimum of the Gibbs potential the second coefficient must also be positive, $G_2 > 0$, however the Weiss mean field analysis shows that below the critical temperature a ferromagnet develops three solutions in the magnetisation, two stable and one unstable which must correspond to minima and maxima in the Gibbs potential. This indicates that below the critical temperature the second coefficient must be negative and hence it can be expanded as a function of $(T - T_C)$,

$$G_2 = (T - T_C)G_2^0 + O(T - T_C)^2 + \ldots \tag{41}$$

so that the coefficient of G_2 changes sign at $T = T_C$ and it is positive for $T > T_C$ and negative for $T < T_C$ such that the high temperature parabolic minimum develops a double minimum separated by a small maximum at $T < T_C$ as required. Application to the ferromagnet in zero field used in the mean-field calculation above shows that for $T > T_C$ there is a single minimum in the Gibbs potential at $M = 0$ and for $T < T_C$ two minima appear, one for a positive M and one for a negative M, with an unstable solution ($\frac{\partial G}{\partial M} = 0, \frac{\partial^2 G}{\partial M^2} < 0$) at $M = 0$ just as in the Weiss mean-field calculation.

Although the Landau theory of phase transitions is a very useful approxima-

tion there is a significant flaw in the assumptions necessary to create the theory. As mean-field theories both the Weiss model of ferromagnetism and the Landau theory of phase transitions assume that fluctuations from the average value of the components of the system, in this case spins, are negligible. In fact this is only a reasonable assumption in regions of phase space far from phase transitions and as a phase transition is approached fluctuations become larger and larger and their effects can dominate the system. An often quoted example of this is critical opalescence. At temperatures close to the critical temperature in a liquid-gas transition the fluctuations in size of the liquid and gas regions reach lengths on the same scale as the wavelength of light and can in theory reach much larger length scales. These fluctuating regions are large enough to scatter light and create an 'opalescence' in what was previously a clear substance which disappears again upon crossing the phase boundary. This behaviour highlights the idea of a divergent length scale near to a phase transition which will be revisited in chapter (6). The characteristic length in a liquid or gas phase system is of the order of nanometers corresponding to the atomic distances between constituent molecules or atoms, but in order to scatter light which has a wavelength of hundreds of nanometers this length must diverge considerably before returning to its original scale on the other side of the transition.

1.3.4 Fluctuations, Critical Exponents and Finite Size Scaling

The Gibbs energy for a ferromagnet, equation (40), has the magnetisation as the order parameter, $\phi = M$. The expansion of the second coefficient, equation (41), quickly shows when minimising the Gibbs energy with respect to the magnetisation there is either a single solution ($M = 0, T > T_C$) or three solutions where the finite solutions are $M \propto (T_C - T)^{\frac{1}{2}}$. Generally the order parameter of a system near to a transition is described by $\phi \propto |t|^{\beta}$ where $t = \dfrac{T - T_C}{T_C}$ is the reduced temperature and is a measure of proximity to the transition and in Landau's mean field treatment of continuous phase transitions the critical exponent governing the order parameter is $\beta = \frac{1}{2}$. Typically six different critical exponents are defined that control the behaviour of different thermodynamic parameters but only two are

Table 2: Thermodynamic properties of a ferromagnet with their corresponding critical exponent. Here d is the lattice dimensionality and $h = \beta H$ is the reduced field.

Thermodynamic property	Critical exponent
Zero field magnetisation	$M \propto \lvert t \rvert^{\beta}$
Zero field specific heat	$C \propto \lvert t \rvert^{-\alpha}$
Zero field susceptibility	$\chi \propto \lvert t \rvert^{-\gamma}$
Critical isotherm	$M \propto \lvert b \rvert^{\frac{1}{\delta}}$
Correlation length	$\xi \propto \lvert t \rvert^{-\nu}$
Pair correlation function at T_C	$G(r) \propto \frac{1}{r^{d-2+\eta}}$

Table 3: Scaling laws relating the critical exponents. Here d is the dimensionality of space.

Scaling law	Due to
$\alpha + 2\beta + \gamma = 2$	Rushbrooke
$\gamma = \nu(2 - \eta)$	Fisher
$\nu d = 2 - \alpha$	Josephson
$\gamma = \beta(\delta - 1)$	Widom

independent due to scaling laws, table (2) shows the critical exponents for magnetic systems and table (3) the scaling laws that relate them to each other. It should be noted that although the critical exponents are often written as functions of the magnitude of the reduced temperature this is not always a valid assumption. Using the magnitude implies that the behaviour of divergent quantities is symmetric about the transition, and although this is often true, the kagome ice model referred to later exhibits divergences from above the transition only hence we should strictly define a pair of critical exponents to cover the behaviour above and below the transition.

The fact that the thermodynamic parameters are governed by power laws is significant as it is a signature of scale invariance. This implies a fractal behaviour that is the same regardless of the scale on which it is examined and it is this fact that accounts for the unusual ability of fluctuations to occur on all length scales near to the transition. The realisation that the behaviour of systems is scale invariant due to the critical exponents near to transitions highlighted the failure of mean-field approaches to describe real systems.

Guggenheim observed that after appropriately normalising the data, measure-

ments of the coexistence curves of eight differing fluids fell onto almost exactly the same curve and that the curve was best fitted using the critical exponent $\beta \approx \frac{1}{3}$ [12]. This, in combination with the influence of the critical exponents, decisively removed Landau theory as an accurate description of phase transitions.

It is one of the more incredible discoveries in physics that thermodynamic properties near to a phase transition are remarkably very similar for many quite different systems. This phenomenon is known as universality and is a consequence of the fact that the microscopic details of a system near to a phase transition are not important, rather it is the macroscopic details that will govern the behaviour [13].

In the critical region near to a phase transition systems are particularly sensitive to their size which leads to effects such as the critical opalescence described above. In that case the onset of the unusual behaviour was at the point where the diverging length scale within the system, the correlation length, became significantly larger than the typical length scale of the system; this criterion can be widely used to define the critical region. Once inside this region then the various critical phenomena of a system all scale to the divergent correlation length rather than a characteristic length of that system and therefore they all scale to each other. This is the central result of renormalisation group theory referred to as the scaling hypothesis. In particular it states that the dominant term in the thermodynamic potential that is appropriate to the transition in the region of the critical point has a form [1],

$$\mathcal{G} \sim |T - T_C|^{2-\alpha} f^{\pm}\left(\frac{H^{1+\frac{1}{\delta}}}{|T - T_C|^{2-\alpha}}\right), \quad T \to T_C \tag{42}$$

which is expressed with the magnetic field as this is relevant to magnetic systems but this is generally the intensive parameter that is conjugate to the order parameter of the system. The function $f^{\pm}$ is called the scaling function and is discontinuous across the transition with the different forms referred to as f^{+} for $T > T_C$ and vice versa. An explicit example of a finite size scaling function is given in chapter (6) in the analysis of the magnetic susceptibility at the Kasteleyn transition.

1.4 Lattice Models

Magnetic lattice models can be viewed simply as a number of spins arranged in a regular manner on a lattice with a possible interaction between the spins. Each of these three components of a model, the lattice, the spins and the interactions, can be defined in many ways, and the many different combinations allow lattice models the flexibility to represent systems with a wide range of properties.

The variation in each of the three components is governed in general by their allowed dimensionality. Magnetic spins can be classified into three types, Ising, XY and Heisenberg, due to the symmetry properties, or the restrictions on the orientation, of the spins. Ising spins are confined to a single dimension and may point in one of only two directions. XY spins are confined to two dimensions and may point in any direction within that plane. Heisenberg spins are three dimensional and may take any direction within a sphere. Similarly we can separate the lattice structures by dimension. The simplest case is 1D when the spins must be arranged in a line, in 2D the spins can be given any regular connectivity within a plane, in 3D the spins can be given any regular connectivity within a 3D Euclidean space and theoretical models have the ability to extend this to an nD space which is a situation where the mean field theory approach is particularly suitable. Finally the interactions between the spins on the lattice have a dimensionality defined by the lattice, but may be short ranged including only one or at most a few neighbouring spins, or long ranged with the general form,

$$E \sim \frac{1}{r^{\sigma}}, \quad \sigma < d - 1 \tag{43}$$

where d is the dimensionality of the lattice.

The spin ice model [14] is central to this thesis however before beginning work on this model some of the simpler models with known solutions are reviewed. These models provide some insight into the complex behaviour of their more detailed relatives and are useful for examining some of the features common to many magnetic systems and their lattice models. After the preliminary models the spin ice model

is treated, gradually confining the dimensionality of the system from 3D to 1D.

1.4.1 Preliminary Models

The Ferromagnetic Ising Square Lattice Model One of the great successes of magnetic lattice models is their use in studying phase transitions and one of the greatest successes in this area is Onsager's solution of the square lattice ferromagnetic Ising model [15]. He was able to show that a system of Ising spins situated on the vertices of a square lattice interacting ferromagnetically with their nearest neighbours only will undergo a phase transition observable as the spontaneous onset of a magnetic moment at a critical temperature given by $\exp(\frac{4J}{k_B T_C}) = 3+2\sqrt{2}$. Also, his model predicted that the specific heat would diverge at the transition rather than simply exhibit a discontinuity as predicted by mean field theory. This result showed that mean field theory was certainly incomplete and encouraged the investigation of critical exponents as described previously. Figure (4) shows the magnetisation from simulations of this model compared to an analytic solution calculated by Yang [16]. This first simulation model highlights the effects of finite size on thermodynamic properties when compared to analytic results which are calculated for a system considered to be infinitely large (in the thermodynamic limit the number of components in a system $N \to \infty$). The analytic solution shows the magnetisation increasing from zero exactly at $T = T_C$ and reaching a maximum value of one at a temperature equivalent to the interaction strength, J. In a finite system there is a sharp increase in the magnetisation centred on T_C but the magnetisation begins to increase at temperatures slightly above T_C and the transition appears 'rounded'.

The Hamiltonian for the model is,

$$\mathcal{H} = -J \sum_{i,j(n.n.)} \mathbf{S}_i \cdot \mathbf{S}_j \tag{44}$$

where the sum runs over all nearest neighbouring, $(n.n.)$, spins in the lattice. This type of interaction between spins is called the exchange interaction and describes almost all of the models in this work. At this point it is important to draw the distinction between the real spins found in a magnetic material and pseudo spins

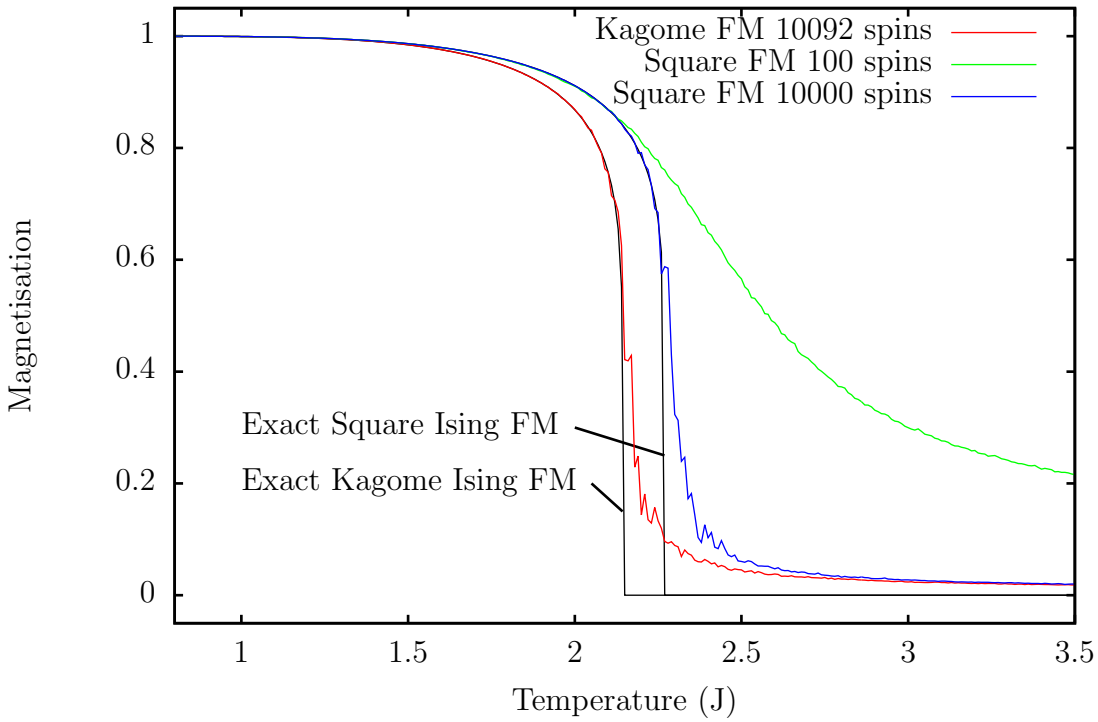

Figure 4: Magnetisation data for the square lattice and kagome lattice with nearest neighbour, ferromagnetically interacting Ising spins. The simulations were performed using 100 and 10000 spins on the square lattice and 10092 spins on the kagome lattice and are also compared to analytic solutions [16, 17]. As the system size is increased the curves tend to their corresponding exact expressions. Fluctuations can be expected to have a magnitude $\sim \mathcal{O}(\frac{1}{\sqrt{N}})$ for a system of N spins, thus the data for the 100 spin simulation is approximately ten times further from zero above the transition than the data from the 10000 spin simulation. The connectivity of the square and kagome lattices is the same however the transition temperature of the ferromagnetic kagome lattice is below that of the square lattice due to the different symmetry in the lattices [18].

which are often used in theoretical models. Real spins have a magnitude that is determined by their spin and orbital angular momenta as described in section (1.1) whereas pseudo spins are employed to simplify the modelling of real magnets and have a magnitude $\sigma = \pm 1$. In this case the correct description of a real spin at site i is,

$$\mathcal{S}_i = \sigma_i \mathbf{d}_{\kappa(i)} M_i^{sp} \tag{45}$$

where $\sigma_i = \pm 1$, $\mathbf{d}_{\kappa(i)}$ is a unit vector in the direction of the real spin and M_i^{sp} is the magnitude of the real spin angular momentum. Throughout this work we shall not be interested in the real spin magnitude and employ the variable $\mathbf{S}$ to represent the unit length spins of the system where,

$$\mathbf{S}_i = \frac{\mathcal{S}_i}{M_i^{sp}} \tag{46}$$

We will then discuss the spins in the system which shall be taken to refer to $\mathbf{S}$ and the pseudo spins which shall be taken to refer to only σ.

The thermodynamic limit is intended to represent experimental reality in mathematical models and generally it is reasonable to assume $10^{-23} \approx 0$, however as the thermodynamic limit is never actually attained there may be some circumstances when it is reasonable to question its use. The critical opalescence mentioned previously was ascribed to fluctuations in the length of the regions of liquid and gas becoming larger as the transition temperature was approached and being able to reach the extent of the system at the transition. There is a length, called the correlation length, within systems that undergo a phase transition and develop an order parameter which is a measure of the extent to which regions possessing that order parameter are correlated, for example the length of the liquid region. The correlation length is divergent at the transition and defined,

$$\xi \propto |T - T_C|^{-\nu} \tag{47}$$

where ν is the correlation length critical exponent defined in table (2). Landau theory predicts the value $\nu = \frac{1}{2}$ however the 3D Ising model predicts $\nu = 0.6298$ and

the 3D Heisenberg model predicts $\nu = 0.7112$ so that a reasonable approximation is $\nu = \frac{2}{3}$ [19]. For a system to obey the predictions made using the critical exponents it is necessary for the correlation length to diverge near to the transition and to be smaller than the largest length in the system such that the correlation length is unrestricted as if it were in an infinitely large system and hence a useful indication of when the thermodynamic limit is relevant is to compare the correlation length to the system size.

Following Goldenfeld [3], a conservative assumption for an approximation to the correlation length is $\xi \approx \xi_0 t^{-\frac{2}{3}}$ where $\xi_0 \approx 1$ nm and t is the reduced temperature. Given a system of size $L = 1$ cm, then $\xi = L$ at $t \approx 10^{-11}$ hence for real materials of macroscopic sizes the assumption of the thermodynamic limit is valid as the correlation length is only constrained to the system length at temperatures so close to the phase transition as to be unmeasurable. In computational models, and recently in nanofabricated materials such as those presented in chapter (7), L may be much closer to ξ_0. If $L = 1000\xi_0$ then at $t \approx 10^{-5}$ the behaviour of the system departs from the theory because the correlation length is predicted to be larger than the system size which is an unattainable situation. For systems of this size and smaller it may be possible to observe the departure of the behaviour from the analytic predictions close to the transition and the thermodynamic limit is not a reasonable assumption. Thus finite size effects must be considered carefully when using computational models as the extent of the model means they will be relevant over noticeable regions of phase space.

The Ising Kagome Lattice Onsager's successful transfer matrix solution of the square lattice Ising model prompted attempts to find analytic solutions to the ferromagnetic Ising model on other regular 2D tilings. Husimi and Syôzi considered the triangular and honeycomb lattices finding similar behaviour in both cases involving a second order phase transition to a low temperature long-range ordered state [20, 21]. A related lattice that was also treated was the kagome lattice; this is a corner sharing arrangement of triangles decorating an underlying triangular Bravais lattice, shown in figure (5).

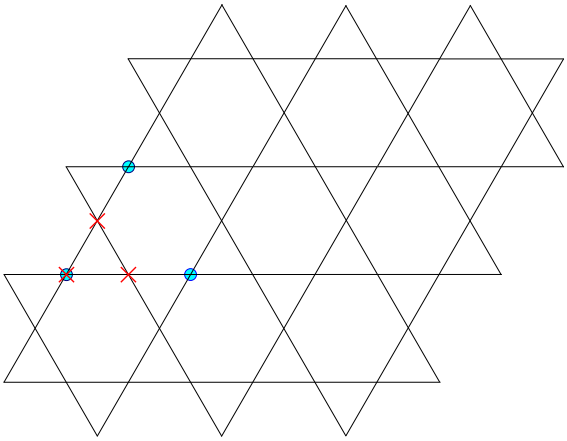

Figure 5: The kagome lattice structure. The lattice is composed of a triangular motif (red crosses) decorating a triangular Bravais lattice (blue circles). The triangles of the motif are one quarter of the size of the underlying Bravais triangles thus an additional triangle is defined by the connections between each of the motifs.

The kagome lattice is similar to the square lattice in that each spin or lattice site has four neighbours and periodic boundary conditions reduce both lattices to a torus; however they are distinct as the kagome lattice is not a Bravais lattice and whilst the square lattice has a four-fold rotational symmetry that of the kagome lattice is six-fold. With ferromagnetically interacting Ising spins constrained to one of the cubic lattice directions the thermodynamic behaviour of the lattice is very similar to the Ising ferromagnetic square lattice model; in the first paper to use the term 'kagome lattice', Syôzi showed that the transition temperature was not the same however and is given by $\exp\left(\frac{4J}{k_BT}\right) = 3 + 2\sqrt{3}$ [18]. The critical temperature of the Ising model on all the Archimedian tilings of the Euclidean plane along with their dual lattices has now been calculated [22]. The exact magnetisation of the ferromagnetic Ising kagome lattice was first derived by Naya following the work of Yang and then Potts on the square and triangular lattices [17].

Wannier treated the triangular and hexagonal lattices with antiferromagnetic Ising spins [23] in what is often quoted as the first investigation into a frustrated magnetic system and Kanô and Naya treated the antiferromagnetic kagome lattice [24]. The introduction of antiferromagnetic coupling between the spins generates frustration within the lattice and is of particular interest as this model maps onto the Wills ice model (described later). Comparison of the energy data for the ferro-

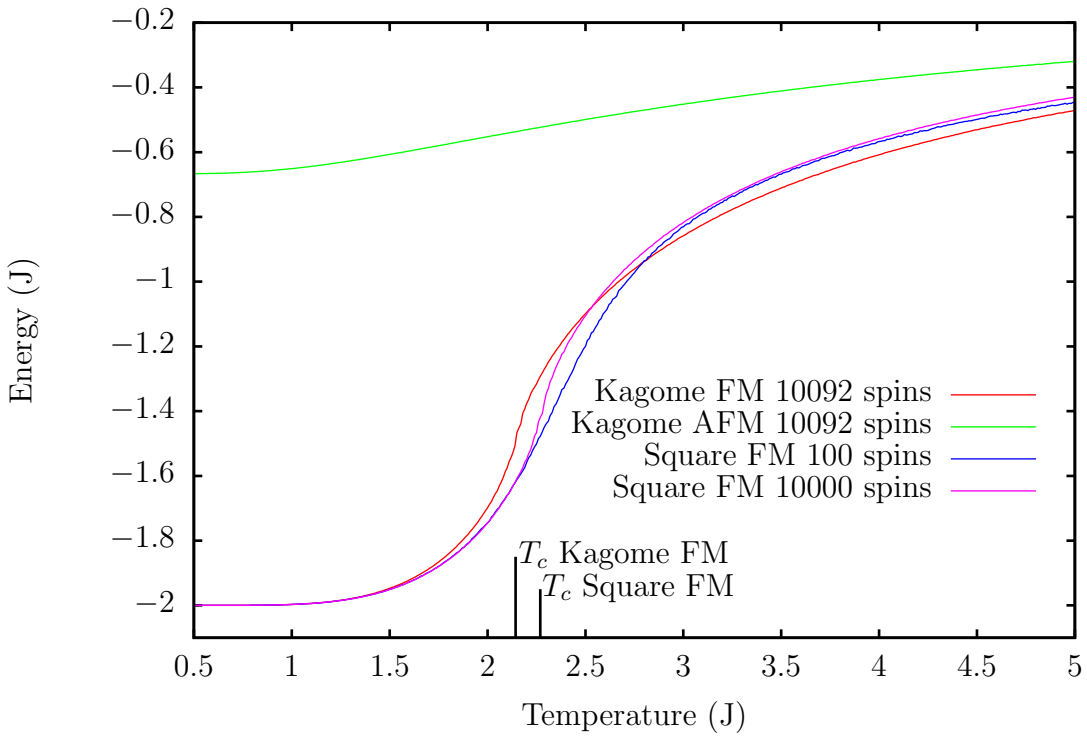

Figure 6: Energy as a function of temperature for simulations of 100 and 10000 Ising spins on the square lattice and 10092 Ising spins on the kagome lattice with ferromagnetic and antiferromagnetic interactions between spins. The number of neighbours is the same for a spin on the square or kagome lattice hence the low temperature energy is identical in the ferromagnetic case, however the differing symmetry generates a slightly different transition temperature. The antiferromagnetic Ising kagome lattice does not undergo a transition to a long range ordered state because it is geometrically frustrated which is reflected in its comparatively higher minimum energy.

magnetic and antiferromagnetic models, figure (6), highlights the effect that frustration has in 'lifting' the level of the ground state energy above the unfrustrated analogues. Figure (7) shows the specific heat and entropy of the frustrated lattice calculated using,

$$C = \frac{N}{T^2}\left(\langle E^2\rangle - \langle E\rangle^2\right) \tag{48}$$

$$S = \int_0^\infty \frac{C}{T} dT \tag{49}$$

and indicates the same high temperature limit of $S_{mag}\,(T \to \infty) \to 0.1948$ JK^{-1}site^{-1} calculated by Wills *et al* [25]. Given that an unfrustrated Ising system must have a high temperature entropy $S_{max} = k_B \ln 2$ JK^{-1}site^{-1} as at high temperatures each spin is not restricted by interactions with others and therefore has two possible orientations only, this value can then be used to calculate the residual entropy due to the frustration,

$$S_{mag}\,(T \to \infty) = \int_0^\infty \frac{C}{T} dT \to 0.1948 \tag{50}$$

$$\implies S_{res} = S_{max} - S_{mag}(T \to \infty) \tag{51}$$

$$= k_B(\ln 2 - 0.1948) = 0.4983 k_B \text{ JK}^{-1}\text{site}^{-1} \tag{52}$$

in good comparison with the residual entropy calculated for the Bethe lattice, $S_{res} = 0.5014 k_B$ JK^{-1}site^{-1} [25]. The calculation of the residual entropy above indicates its meaning: it is the entropy remaining at $T = 0$. A simple version of the third law of thermodynamics states that the entropy of the system must tend to zero as the temperature tends to zero but in frustrated materials there is a finite residual entropy. This apparent violation of the law requires it to be restated as the entropy of a perfect crystal must tend to zero as the temperature tends to zero. Geometrically frustrated models may retain a finite residual entropy because there is no unique groundstate for them to reach as the temperature decreases and the statistical mechanical definition of the entropy dictates that the entropy is proportional

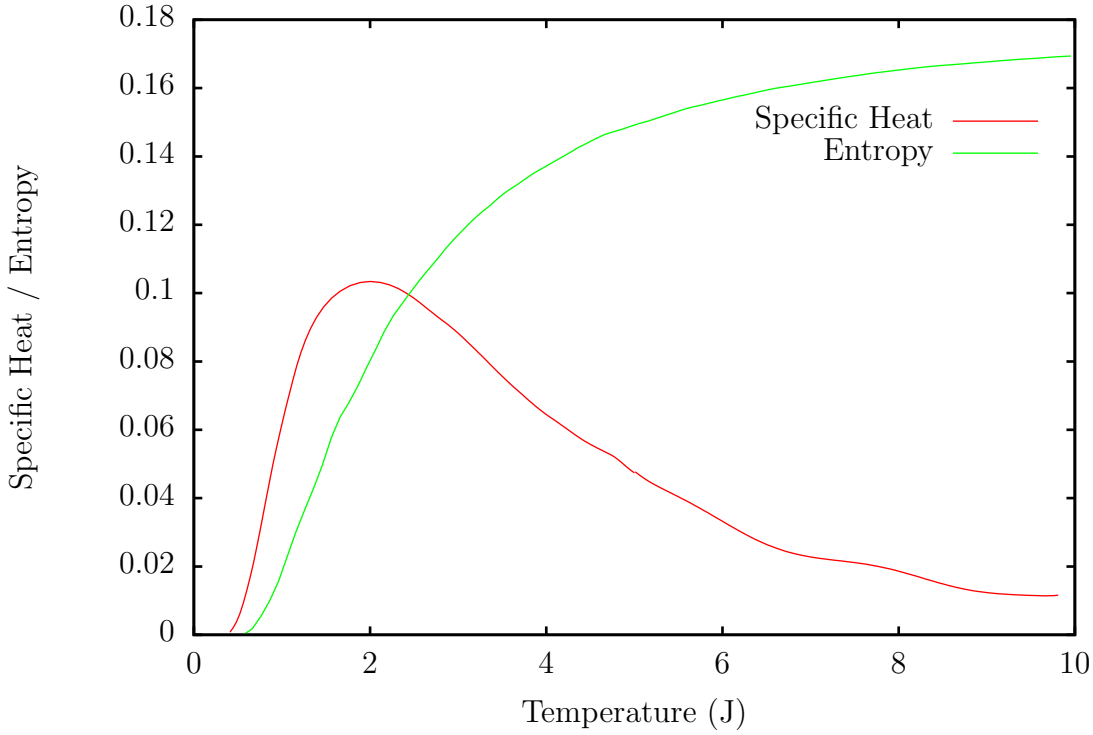

Figure 7: Intensive (per spin) specific heat and entropy from simulation of 10092 spins on the kagome lattice with antiferromagnetic interactions. The geometric frustration in this lattice creates a macroscopically degenerate groundstate which prevents the integrated entropy achieving its maximum value of $S = k_B \ln 2 = 0.693 k_B$ JK^{-1}site^{-1} The difference between the maximum attained value and the unfrustrated maximum is known as the residual entropy.

to the number of states inhabited by the system.

$$S = k_B \ln \Omega \tag{53}$$

Geometrically frustrated systems often have macroscopically degenerate groundstates, $\Omega \gg 1$, and hence a large residual entropy as in the case of the spin ice model.

1.4.2 3D - Spin Ice

The magnetic lattice model central to this thesis is spin ice [14]. The model is described from a theoretical perspective in the following text and revisited in chapter (2) where it is approached from an experimental point of view.

Anderson's work on pyrochlore lattices with antiferromagnetic Ising spins constrained to the cubic directions of the lattice [26] is regarded as some of the earliest

work on frustration, before the term had first been used to describe such a situation by Toulouse in his work on spin glasses [27], and the pyrochlore lattice is certainly one of the most well studied of the magnetic lattice models; however, before the spin ice model the focus was on antiferromagnetic spins. Spin ice stands out in the history of frustrated magnetic lattice models for being the first to exhibit frustration with ferromagnetic interactions between spins.

The spin ice model has Ising spins similarly residing on the vertices of the pyrochlore lattice but constrained to lie along the body centred diagonal directions of each tetrahedron, see figure (8). The spins interact ferromagnetically under the topological constraint that the divergence of the magnetic field in any tetrahedron must be zero,

$$\nabla \cdot \mathbf{M}(\mathbf{r}) = 0 \tag{54}$$

This constraint has far reaching consequences and will be returned to throughout this thesis. It is realised by ensuring that of the four spins on a tetrahedron there are always two spins pointing in and two spins pointing out, however of the $2^4 = 16$ possible spin configurations on a tetrahedron six satisfy this constraint creating a degeneracy in the ground state of each tetrahedron. Further, as the tetrahedra share corners only, the configuration on any tetrahedron does not uniquely determine that of its neighbours and the degeneracy in the ground state of the lattice is extensive; experimental evidence of this is presented in chapter (2). The inability of the lattice to uniquely order is a signature of the frustration present; this is also in apparent contradiction to the third law of thermodynamics and identifies a characteristic zero point entropy associated with frustrated lattices as described above in the context of the kagome lattice. The name spin ice is due to the exact mapping between this model and water ice, identified by Harris and Bramwell [14, 28] which can also be seen through the simultaneous mapping of both to the antiferromagnetic Ising pyrochlore lattice model which Anderson originally considered [26].

The spin ice model was conceived with an effective ferromagnetic nearest neigh-

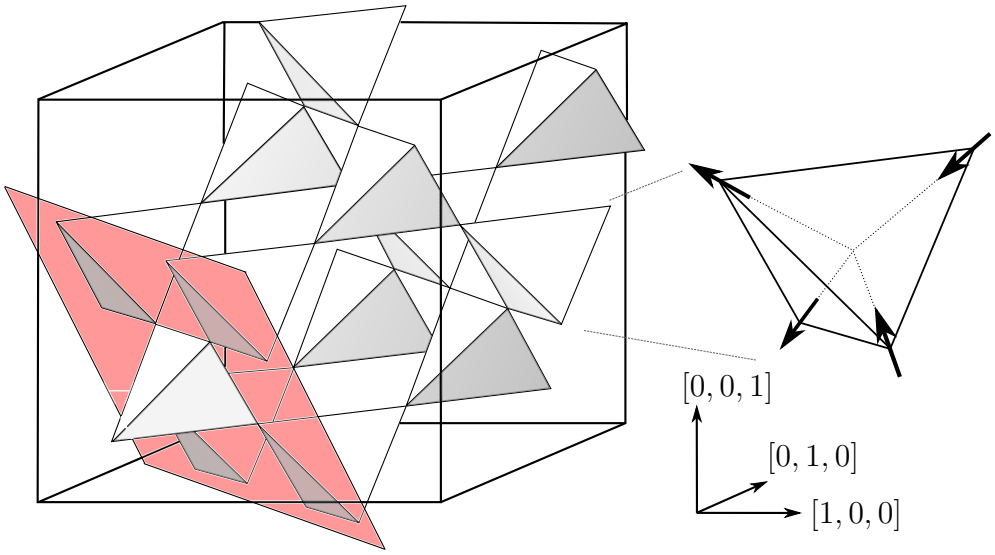

Figure 8: The Pyrochlore lattice of the spin ice model with one tetrahedron highlighted illustrating the topological constraint that maintains a divergence free magnetic field overall and within each tetrahedron. Each spin is restricted to the $[1, 1, 1]$-type body centred diagonal directions indicated with dotted lines inside the tetrahedron. The pink plane highlights one of the planes through the lattice that define the kagome lattice referred to in the kagome ice model.

bour exchange coupling, J, and modelled with the Hamiltonian,

$$\mathcal{H} = -J \sum_{i,j(n.n.)} \mathbf{S}_i \cdot \mathbf{S}_j - \mathbf{H} \cdot \sum_i \mathbf{S}_i \tag{55}$$

where $\mathbf{H}$ is an externally applied magnetic field.

There have been significant efforts since the inception of the model to develop a Hamiltonian that encompasses a more complete range of interactions and a sufficient expression must include at least an antiferromagnetic nearest neighbour exchange coupling, J, a long range dipolar interaction of magnitude D, and a strong easy axis anisotropy term, $E \ll 0$ [29–33] .

$$\mathcal{H} = -J \sum_{i,j(n.n.)} \mathbf{S}_i \cdot \mathbf{S}_j + D r_{n.n.}^3 \sum_{j>i} \frac{\mathbf{S}_i \cdot \mathbf{S}_j}{|\mathbf{r}_{ij}|^3} - \frac{3(\mathbf{S}_i \cdot \mathbf{r}_{ij})(\mathbf{S}_j \cdot \mathbf{r}_{ij})}{|\mathbf{r}_{ij}|^5} + E \sum_i (\mathbf{d}_{k(i)} \cdot \mathbf{S}_i)^2 - g\mu_B J \mathbf{H} \cdot \sum_i \mathbf{S}_i \tag{56}$$

where

$$r_{n.n.} = \text{Nearest neighbour spin distance.}$$

$$\mathbf{S}_i = \text{Spin on lattice site } i.$$

$$\mathbf{r}_{ij} = \text{Position vector between spin } i \text{ and } j \text{ measured in terms of } r_{n.n.}.$$

$$\mathbf{d}_{k(i)} = \text{Easy axis unit vector at site } i.$$

and the easy axes are the body centred directions of the tetrahedron

$$\mathbf{d}_0 = (1, 1, 1)$$

$$\mathbf{d}_1 = (\bar{1}, \bar{1}, 1)$$

$$\mathbf{d}_2 = (1, \bar{1}, \bar{1})$$

$$\mathbf{d}_3 = (\bar{1}, 1, \bar{1})$$

Remarkably the net effect of the antiferromagnetic exchange and the dipolar interaction is equivalent to an effective ferromagnetic exchange term and allows a Hamiltonian such as equation (55) operating on Ising pseudo-spins confined to the easy axes of the pyrochlore lattice to reproduce the behaviour of almost all the accessible states of the full Hamiltonian [30, 34, 35]. In short, the behaviour of spin ice is governed by a topological constraint, equation (54), commonly referred to as the ice rules following the correspondence with water ice and the proton ordering rules developed by Bernal and Fowler [36].

Spin ice couples to an external magnetic field in an interesting way as it induces a reduced dimensionality within the model. It was quickly observed [14] that the application of a magnetic field in certain directions lifts the degeneracy within the model and can lead to an ordering transition. The application of a magnetic field along the [111] direction leads to a more complex, two-stage, ordering process which is readily observed as a double plateau in the magnetisation-field curve [37]. Whilst the system is in the regime of the first plateau it can be considered as a layered

lattice structure with alternating triangular and kagome geometries, see figure (9)[1]. The spins on the layers with triangular lattice geometry are static with respect to the energy scale of the applied field and completely ordered and so provide a form of 'insulation' between adjacent kagome layers where the spins retain a degenerate groundstate; it is therefore a good approximation to consider the kagome layers as individual, two-dimensional lattices.

Application of a magnetic field in a suitable direction parallel to the kagome plane can then invoke the emergence of 'chains' of spins, that is spins arranged in a head-to-toe fashion stretching for a considerable distance across the lattice. The extent of this dimensional reduction, as with the previous one, is tunable with the applied magnetic field and tilting a [111] field slightly toward the [110] direction can produce chains with a length on the order of the lattice edge. In this scenario these chains may be considered as one-dimensional and so, within the three-dimensional spin ice model, there are also examples of two- and one-dimensional magnetic structures that are easily accessible. The ease with which an applied magnetic field can alter the spin ice lattice is one of its great strengths and a recurring theme of this work.

1.4.3 2D - Kagome Ice

The application of a magnetic field in the [111] direction to the spin ice model isolates consecutive triangular and kagome lattice planes as illustrated in figure (9). Within a suitable range of field strengths one of the four apical spins on each tetrahedron within the pyrochlore lattice is pinned to the [111] direction as this spin is parallel to the field and together these spins form the triangular lattice planes whilst the remaining three spins maintain the 'two in, two out' ice rule and form the kagome lattice planes. The triangles in the kagome plane alternate between having apical spins above or below their lattice plane and hence the spins on the kagome triangles alternate between two spins pointing into and one out of, or two spins pointing out of and one into each triangle. Due to the corner sharing nature of the pyrochlore lattice each of the triangles with an apical spin below the plane is completely defined by the surrounding triangles with apical spins above the plane; henceforth triangles

[1]A portion of figure(9) is courtesy of Tom Fennell.

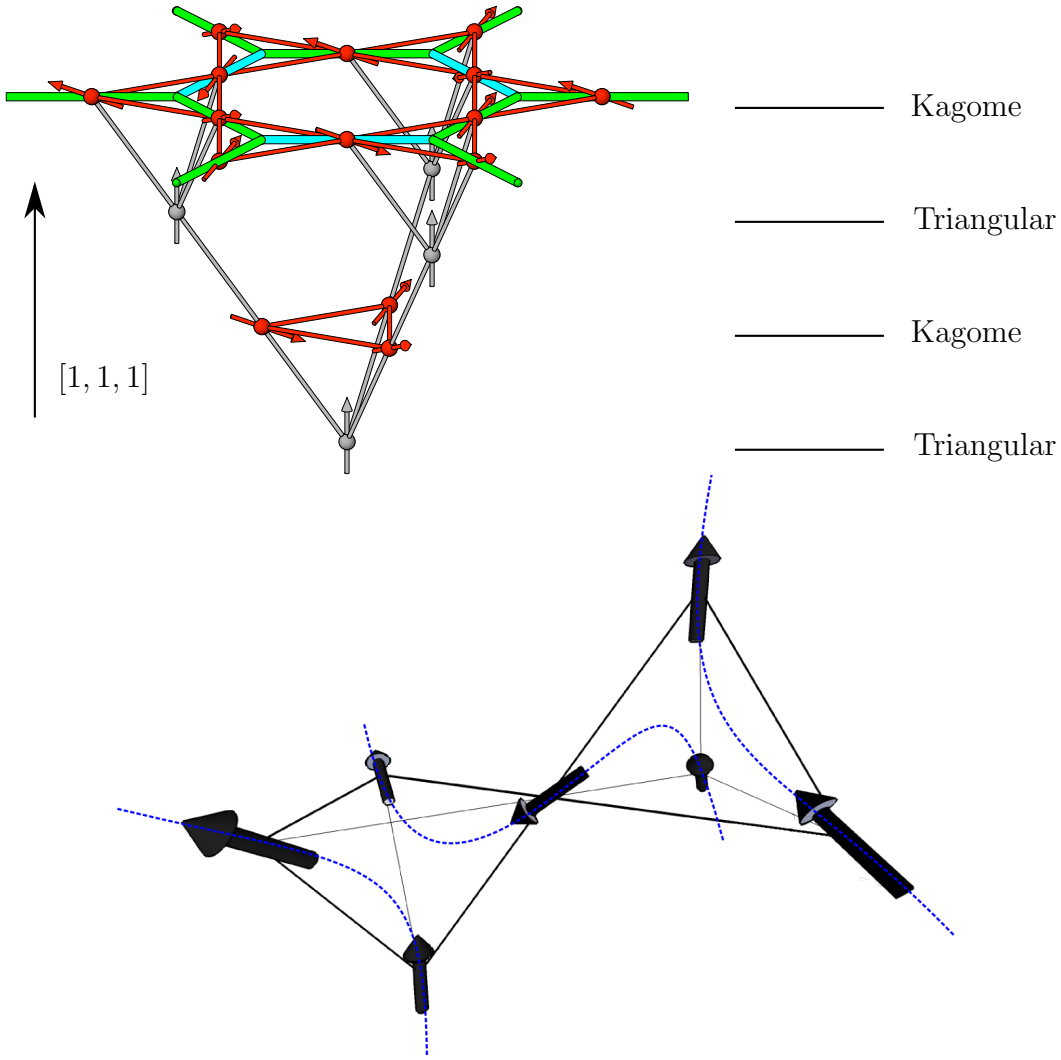

Figure 9: Top: The application of a magnetic field in the [111] direction to the spin ice model isolates layers of spins with consecutive triangular (gray spins) and kagome (red spins) geometries. The bottom layer has triangular geometry and the consecutive kagome and triangular layers above that show the offset in the way they are stacked. The top kagome layer indicates the mapping to the dimer representation of the kagome lattice where the dimer (light blue) corresponds to the outward pointing spin on an up triangle. Bottom: An up triangle (right) and a down triangle (left) defined by whether the pinned spin in the triangular layer is above or below the kagome triangle. The spins on the kagome plane remain on the body-centred directions and so are canted into and out of the plane. The dotted blue lines indicate the magnetic field which obeys the divergence free condition, equation (54).

with an apical spin above the kagome plane will be referred to as 'up' triangles and those with an apical spin below the plane as 'down' triangles respectively as illustrated by figure (9).

The ice rule on the spin ice lattice can be interpreted in a continuum description as a divergence free field (equation (54)); however the same ice rule on the kagome spin ice lattice does not produce a divergence free field constraint for the spins as there are now two in and one out. It is possible to redefine the weight of the spins so that for a triangle with two spins pointing in and one out the outward spin has twice the value of the inward spins. This mapping permits the definition of a suitable divergence free constraint on the kagome lattice only which we shall examine in further detail in chapter (4).

The kagome lattice is also not exactly 2D because the spins making up each triangle in the kagome lattice are canted into or out of the plane of the lattice as they remain on the trigonal directions of the spin ice pyrochlore lattice, further, the three spins making up each triangle are still part of a tetrahedron and remain strongly coupled to the fourth spin which is entirely out of the kagome plane; thus when considering this model it is necessary to include the consequences of the canted spins and the fourth spin and although this is approximately a 2D model this requires the third dimension perpendicular to the plane; kagome ice is therefore only a pseudo-2D model.

The mathematical framework of the model is now defined explicitly, see figure (10), in order to make reference to the lattice and model more clearly. These definitions will be useful when describing neutron scattering for example.

The direct space lattice vectors are defined,

$$\mathbf{a} = (1, 0, 0) \tag{57}$$

$$\mathbf{b} = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right) \tag{58}$$

$$\mathbf{c} = (0, 0, 1) \tag{59}$$

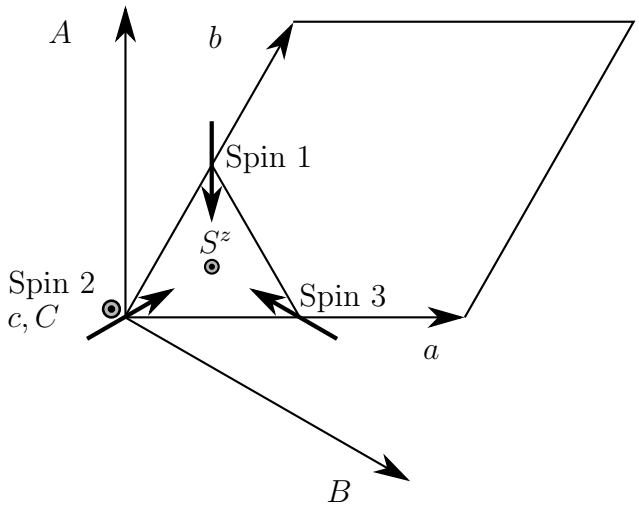

Figure 10: Schematic diagram showing one primitive unit cell of the real space kagome lattice with basis vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$, and their reciprocal lattice vectors, $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. Vectors $\mathbf{c}$ and $\mathbf{C}$ are pointing out of the page. Also shown are the three spin positions (at the midpoints of the arrows) which decorate the Bravais triangular lattice to form the kagome lattice. At each spin position there is an Ising-type spin where the spin direction is defined as positive when pointing into the centre of a triangle as shown. The unit length spins of the spin ice pyrochlore lattice are resolved into a component parallel to the kagome plane and a component parallel to the $[1, 1, 1]$ direction which is the c axis for kagome ice.

The spin positions $(1, 2, 3)$ are defined,

$$\mathbf{r_1} = n\mathbf{a} + \left(m + \frac{1}{2}\right)\mathbf{b} \tag{60}$$

$$\mathbf{r_2} = n\mathbf{a} + m\mathbf{b} \tag{61}$$

$$\mathbf{r_3} = \left(n + \frac{1}{2}\right)\mathbf{a} + m\mathbf{b} \tag{62}$$

where

$$n, m \in \mathbb{Z} \quad [0 : side] \tag{63}$$

The spin components are defined,

$$\mathbf{S_1} = \sigma_1 \left(\frac{2\sqrt{2}}{3}\right)(0, -1, 0) \tag{64}$$

$$\mathbf{S_2} = \sigma_2 \left(\frac{2\sqrt{2}}{3}\right)\left(\frac{\sqrt{3}}{2}, \frac{1}{2}, 0\right) \tag{65}$$

$$\mathbf{S_3} = \sigma_3 \left(\frac{2\sqrt{2}}{3}\right)\left(-\frac{\sqrt{3}}{2}, \frac{1}{2}, 0\right) \tag{66}$$

$$\mathbf{S_z} = \frac{\sigma_z}{3}(0, 0, -1) \tag{67}$$

where

$$\sigma = \pm 1 \tag{68}$$

$\sigma > 0$ means the spin is pointing into the centre of a tetrahedron. On the kagome ice manifold $\sigma_z = -1$ at all times. The basis vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ have unit length in the kagome plane while the spins have unit length in the pyrochlore lattice and remain on its $\langle 111 \rangle$ axes hence they have a projected length onto the kagome plane less than one. Each spin therefore also has a component perpendicular to the kagome plane, along the [111] axis, denoted S_z such that,

$$|\mathbf{S}| = \sqrt{\mathbf{S}^2_{n=1,2,3} + \mathbf{S}^2_z} = 1 \tag{69}$$

and

$$|\mathbf{S}_{n=1,2,3}| = \frac{2\sqrt{2}}{3} \tag{70}$$

$$|\mathbf{S}_z| = \frac{1}{3} \tag{71}$$

Finally the reciprocal lattice vectors are defined,

$$\mathbf{A} = \frac{2\pi \mathbf{b} \times \mathbf{c}}{\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})} \tag{72}$$

$$= \frac{2\pi}{\frac{\sqrt{3}}{2}} \left(\frac{\sqrt{3}}{2}, \frac{-1}{2}, 0 \right) \tag{73}$$

$$= 2\pi \left(1, \frac{-1}{\sqrt{3}}, 0 \right) \tag{74}$$

$$\mathbf{B} = \frac{2\pi \mathbf{c} \times \mathbf{a}}{\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})} \tag{75}$$

$$= \frac{2\pi}{\frac{\sqrt{3}}{2}} (0, 1, 0) \tag{76}$$

$$= 2\pi \left(0, \frac{2}{\sqrt{3}}, 0 \right) \tag{77}$$

$$\mathbf{C} = \frac{2\pi \mathbf{a} \times \mathbf{b}}{\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})} \tag{78}$$

$$= \frac{2\pi}{\frac{\sqrt{3}}{2}} \left(0, 0, \frac{\sqrt{3}}{2} \right) \tag{79}$$

$$= 2\pi (0, 0, 1) \tag{80}$$

Wills Ice A simplification of the kagome ice model is to restrict the dimensionality to strictly 2D in which case it is the model of Wills *et al* [25]. In this case there is no fourth spin lying above or below the lattice plane and the spins are now constrained to the three bisectors of each triangle parallel to the lattice plane. This lattice with antiferromagnetic spins has long been known to be a frustrated system while the ferromagnetic case was discovered to be an extremely frustrated system by Wills *et al.* following the discovery of the ferromagnetic frustrated spin ice system [14, 25] Wills *et al.* refer to this model as *kagomé* spin ice however to avoid confusion with the spin ice and kagome ice lattices already discussed *kagomé* spin ice is henceforth referred to as Wills ice following reference [38]. The ice rule in Wills ice restricts each

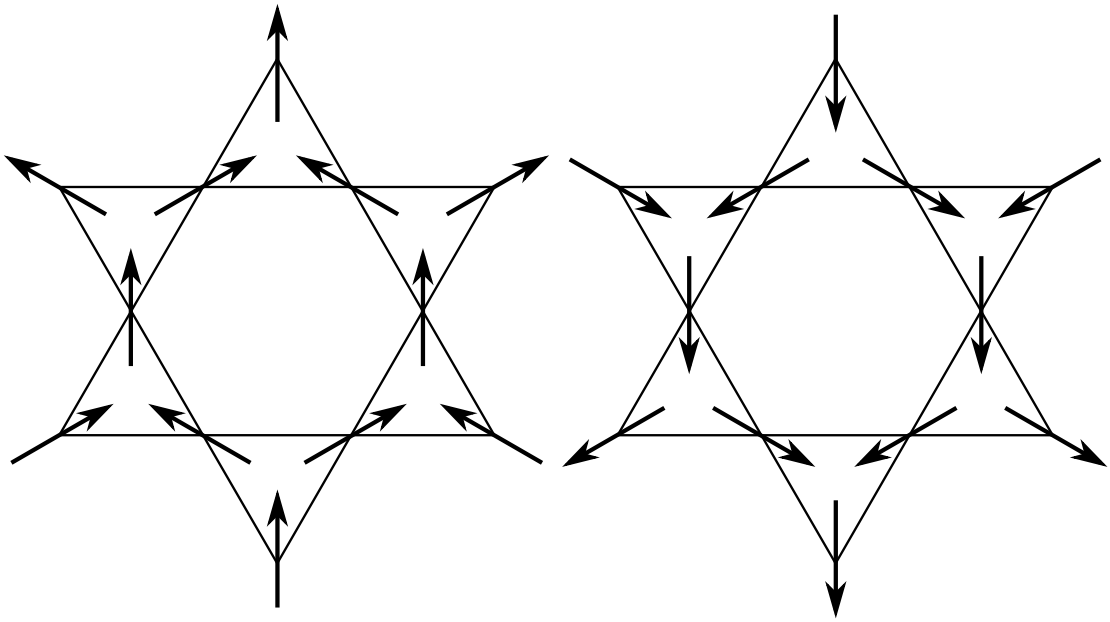

Figure 11: Wills ice allows either of the spin configurations shown and combinations of both but the topological constraints in kagome ice will only allow one of the two. Which of the two is selected depends on whether the pinning field is applied to the pyrochlore lattice in the positive or negative [111] direction and creates a broken topological symmetry in the model.

triangle to either two spins pointing in and one out or the reverse which allows a $\mathbf{Q} = 0$ type ordering with a net magnetisation along any of the three spin sublattice directions *and* along the reverse directions (an example of which is shown in figure (11) along sublattice direction 1). This is in contrast to the kagome ice lattice in which the addition of an extra constraint from the full spin ice rules prevents the formation of $\mathbf{Q} = 0$ order along the reverse directions.

This difference makes Wills ice rotationally six-fold symmetric whilst kagome ice is three-fold. Both models have corresponding mirror symmetries, but whilst Wills ice is symmetric with respect to time reversal, kagome ice has broken time reversal symmetry as the formation of a spin structure corresponding to this condition would violate the topological constraint of the model. The extra constraint that takes the Wills ice model to the kagome ice model is a topological symmetry breaking which is elaborated upon in chapter (4). The lack of time-reversal symmetry breaking in the Wills ice model prevents the appearance of a Kasteleyn transition, however given that Wills *et al.* point out that Wills ice maps onto the Ising antiferromagnetic kagome lattice in which there is no phase transition this is unsurprising.

1.4.4 2D - Square Spin Ice

An alternative approach to an exactly 2D analogue of spin ice can be realised by considering the pyrochlore lattice parallel to the cubic [001] direction. From this perspective the four spins on a tetrahedron appear as a cross, and the pyrochlore lattice appears to be a square lattice, see figure (12). The same topological constraint still applies to this square ice in that on any vertex there must be two spins pointing toward the centre and two pointing away from it, and there are still six of a possible sixteen configurations that obey this rule; hence square ice is frustrated and has a macroscopically degenerate ground state. Dipolar square ice differs from 3D spin ice in that by projecting the tetrahedra onto 2D vertices the long ranged part of the dipolar interaction is not as well screened as in the original model and two of the six ground state vertices are distinguished from the remaining four. Any of the six ground state vertices in spin ice has a net magnetic moment whereas the square ice model has two vertices with spins aligned antiparallel and hence a null moment. This lowers the energy of these vertices such that the sixteen vertices are split into four, rather than three, distinct energy levels in the absence of an external magnetic field.

1.4.5 1D - Spin Chain

Kagome ice and square ice are both composed of chains of spins, in square ice there are two sets at right angles to each other, while in kagome ice there are three sets of chains at 60° to each other. The application of a field parallel to a chain direction will isolate the behaviour of one subset of chains from that of the remaining spins on the lattice. The clearest example of this forms a major component of the work on square ice, that is, by applying a field along one of the spin directions half of the spin chains are totally ordered by the field whilst the other half, perpendicular to the field, are unaffected by it to a first approximation. In square ice the spins are parallel to the direction of the chain, whilst in the kagome case the spins are angled alternately above and below, and to the right and the left of the chain axis but can be considered via their projection onto the chain axis, in which case the resultant

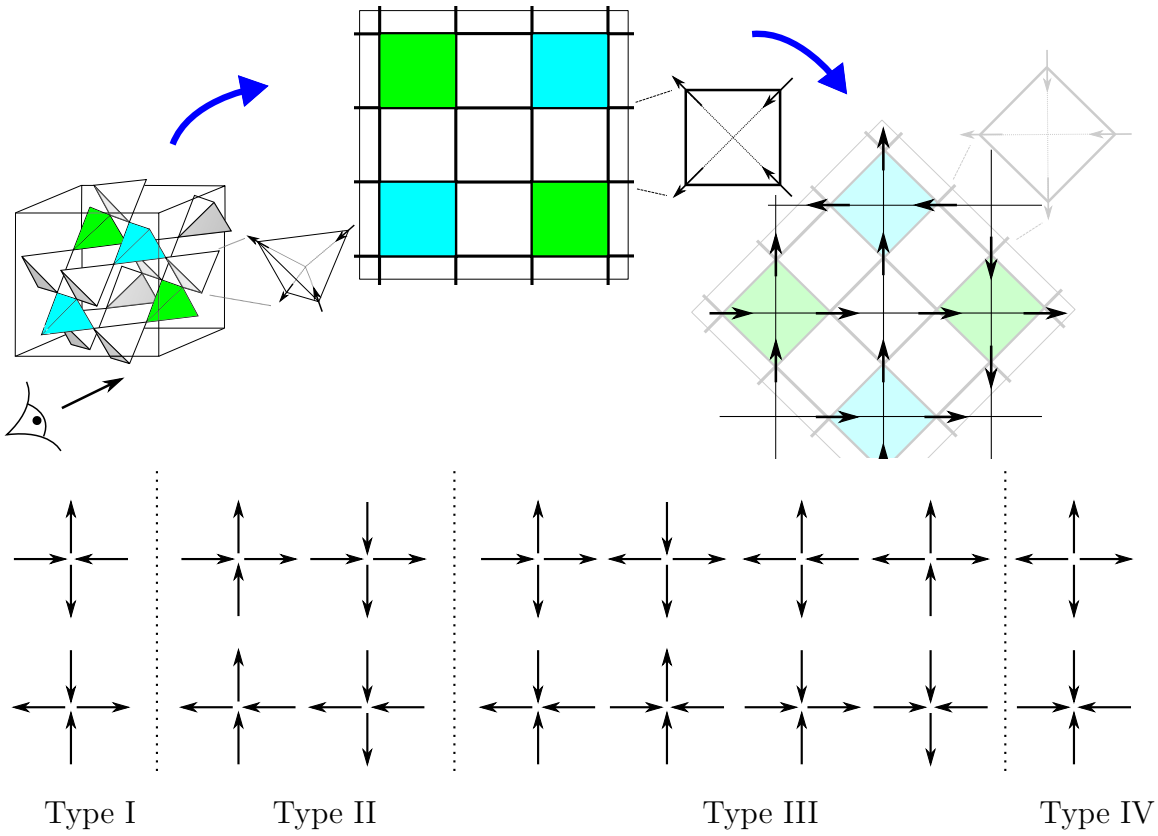

Figure 12: Above: When viewed along one of the cubic axes the pyrochlore lattice appears as a square lattice and the four spins on each tetrahedron project onto a cross shaped vertex. Below: There are sixteen possible spin configurations at a vertex and although six of these vertices respect the two in, two out ice rule as in the spin ice lattice the Type I vertices have a null moment and hence a different energy compared to the Type II vertices. This behaviour is a consequence of the projection of the dipolar interaction onto two dimensions.

chain is equivalent to that of the square ice case with a weaker coupling between spins.

The physics of the 1D Ising model was originally treated by Ising during his doctoral studies with Lenz [39]. The spin chain he considered was slightly different in that the spins were defined as perpendicular to the chain axis, but the results of his work are presented here as they apply to both cases. Ising's solution proceeds via the transfer matrix approach which has been used in many lattice model solutions since, the key results are presented here. Consider a chain of N spins with boundary conditions such that the chain runs from spin $\mathbf{S}_1$ to $\mathbf{S}_N$ and these two spins are neighbours, this produces a finite ring of spins. This situation can be modelled using a Hamiltonian such as [40],

$$\mathcal{H} = -J \sum_{i}^{N} \mathbf{S}_i \cdot \mathbf{S}_{i+1} - \mathbf{H} \cdot \sum_{i}^{N} \mathbf{S}_i \tag{81}$$

where $\mathbf{H}$ is an arbitrary external magnetic field and J is the effective exchange interaction. First we re-scale the interaction parameters with temperature, $K = \beta J$ and $\mathbf{h} = \beta \mathbf{H}$. An expression for the partition function of a finite ring of N Ising spins can be obtained via the trace of the transfer matrix representing the Hamiltonian

$$\mathcal{Z}_N = \sum_{\mathbf{S}_1} \dots \sum_{\mathbf{S}_N} \exp\left(K \sum_{i}^{N} \mathbf{S}_i \cdot \mathbf{S}_{i+1} + \mathbf{h} \cdot \sum_{i}^{N} \mathbf{S}_i \right) \tag{82}$$

$$= \mathrm{Tr}(T^N) \tag{83}$$

where

$$T = \begin{pmatrix} \exp(K+h) & \exp(-K) \\ \exp(-K) & \exp(K-h) \end{pmatrix} \tag{84}$$

and

$$h = |\mathbf{h}| \tag{85}$$

hence

$$\mathcal{Z}_N = \lambda_+^N + \lambda_-^N \tag{86}$$

where $\lambda_\pm$ are the eigenvalues of the transfer matrix

$$\lambda_\pm = \exp(K)\cosh(h) \pm (\exp(2K)(\sinh(h))^2 + \exp(-2K))^{\frac{1}{2}} \tag{87}$$

Generally the derivation proceeds by taking the thermodynamic limit to find that magnetisation is not supported at any finite temperature and observing the critical point and related exponents at $T = 0$. This result led Ising to be disappointed with the model as it did not exhibit a phase transition and he ceased to work any further on it. A more interesting situation arises by considering the consequences of finite size systems [41]. The average magnetisation per spin can be obtained by differentiating the partition function with respect to the re-scaled field,

$$\langle M \rangle(T) = \frac{1}{N}\frac{T}{\mathcal{Z}_N}\left(\frac{\partial \mathcal{Z}_N}{\partial h}\right)_{h\to 0} \tag{88}$$

With no external field this quantity tends to zero as on average the moments of the spins will be equally split between the opposite Ising directions. A quantity of more relevance is the mean square spin excess $\langle M^2 \rangle$ as this does not average to zero.

$$\langle M^2 \rangle(T) = \frac{1}{N^2}\frac{T^2}{\mathcal{Z}_N}\left(\frac{\partial^2 \mathcal{Z}_N}{\partial h^2}\right)_{h\to 0} \tag{89}$$

$$= \frac{1}{N}\frac{1-\tanh(K^N)}{1+\tanh(K^N)}\exp(2K) \tag{90}$$

This quantity tends to 1 at finite T for all values of N however the transition temperature tends to zero with increasing N such that in the thermodynamic limit the transition occurs at $T = 0$ and the chain cannot support ferromagnetism just as Ising concluded. By calculating the magnetisation of the 1D Ising chain as $\sqrt{< M^2 >}$ it is clear that for finite size chains there is a regime controlled by the parameter K in which long range order is supported, and although a phase transition is strictly

only defined in the thermodynamic limit the chain then exhibits at least a crossover from ordered to disordered behaviour at a finite temperature. Simulation results displaying this behaviour are shown in chapter (7).

1.5 Aims of this work

In 1963 Kasteleyn discovered an unusual phase transition in the ordering of hard-core dimers on a hexagonal lattice which is driven entirely by entropic considerations [42]. His transition was conceived within a theoretical model, however several diverse physical systems were later discovered which appeared to exhibit Kasteleyn transitions (see chapter (4) for further detail) until, of particular significance for this work, in 2003 Moessner and Sondhi noted that the Kasteleyn transition was observable in theory in the kagome ice phase of the spin ice model [33].

The discovery of a magnetic model, and an associated physical system, which exhibits a specific phase transition is generally regarded as particularly useful because of the wealth of experimental techniques for magnets and the clean signature of phase transitions in magnetic models; for example the paradigm of the Ising square lattice model with regard to a second order transition. Matsuhira *et al.* had experimentally confirmed the kagome ice state [43] only a year before Moessner and Sondhi's work hence interest was revived in the Kasteleyn phase transition at the same time as scientists working in the field of spin ice were particularly keen to investigate the properties of the kagome ice phase. Spin ice materials are well suited to neutron scattering investigations due to the large moments on the rare earth ions and so Fennell *et al.* employed this technique in an attempt to observe experimental signatures of Kasteleyn physics and provided some of the first validation of Moessner and Sondhi's predictions [44]. Although their work was successful in highlighting the unusual spin correlations (pinch points) associated with the divergence free constraint in spin ice and observing some of the changing spin correlations associated with the Kasteleyn transition it was clear that this system posed many questions and was worthy of further experimental and theoretical investigation.

In this work our intention was to produce a computer simulation model of the

kagome ice phase of spin ice utilising a non-local loop algorithm update with which it would be possible to further explore the Kasteleyn transition under the influence of any temperature or applied field within the kagome plane. The model was designed with two main simulation modes in mind, firstly to produce thermodynamic data as a function of either field or temperature enabling the characterisation of the transition, and secondly to produce lattice spin configurations to be used in calculating neutron scattering intensity maps. This led to a related goal of creating a separate program that can read spin configuration maps and process them into neutron scattering patterns.

With the ability to simulate the effects of arbitrary field and temperature on kagome ice and produce results as both thermodynamic quantities and neutron scattering maps our aim was to make connection to the neutron scattering work performed by Fennell *et al.* in order to better understand their experimental results and also to provide predictions for the conditions necessary to observe the Kasteleyn transition and the scattering patterns observable as it is approached. We intended to use our model to investigate the behaviour of kagome ice when the inplane field driving the Kasteleyn transition was rotated. There had been no experimental or theoretical data produced in this region of phase space until this work and so we aimed to examine the manner in which varying the field angle changed the existing results and hopefully inspire future experimental work to verify our predictions.

The kagome ice research formed the majority of this doctorate but a significant second area of research was carried out on the square ice model. Inspired by an experiment performed on a magnetic square lattice nanoarray which produced novel results in the field of magnetic nanoarrays [45] we aimed to create a model which could provide complementary theory to that data. Given the complex interaction effects in mesoscopic nanoarrays our aim was not to design a model that replicated the experimental nanoarray, rather the simplest model that remained relevant to it so that the results would be a guide to the behaviour displayed by the real system but would remain fully understandable. The aim of the square ice model was to produce thermodynamic data, particularly magnetisation data, that would describe

the experimental data as closely as possible whilst the elements of the model and the interactions between them were very simple, however as the investigation proceeded it became apparent that a theoretical model of a finite size 1D Ising chain of spins provided information on a more general level regarding the behaviour displayed by the experimental system. This prompted us to focus on the finite size Ising chain where we aimed to show a general result regarding the effect of finite size on the magnetisation.

In both cases we aimed to create computer simulation models of complex, frustrated magnetic systems inspired by experimental work which indicated there were unanswered questions in these systems. Our approach has been to formulate models from simple components and interactions so that the results produced are easily understood but remain relevant to experiment. Magnetic systems are satisfying to model as the correlation between experiment, theory and simulation is higher than in many other systems and in this work we have found that to be true. We attempted to make links back to our experimental colleagues at all times and set out to produce theory and simulation that we hope will stimulate future work.

2 Experimental Systems and Measuring Techniques

All of the work in this thesis has been achieved using computational simulations of magnetic lattice models; however this has always been informed by and compared with experimental work. In this chapter we introduce experimental realisations of, and systems related to, the spin ice model and display some of the experimental results obtained by other groups that are relevant to this work. We also introduce the measuring techniques used to obtain these results. In the case of neutron scattering this will be substantially expanded upon in the following chapter, however the Magneto-Optical Kerr Effect (MOKE) is not used in the simulations and will only be briefly described.

2.1 Classical Spin Ice Materials

Geometrically frustrated systems are unable, as a consequence of their structure, to simultaneously minimise the energy associated with each of their bond interactions typically leading to a macroscopically degenerate ground state and a zero-point entropy. The first system in which such geometrical frustration was identified was water ice with interest in this system raised by the observation of a discrepancy between the spectroscopic value of the entropy and that calculated by integrating the specific heat [46]. In that reference Giauque and Stout note that Linus Pauling suggested an explanation for the discrepancy invoking the random ordering of the hydrogen bonds according to the Bernal-Fowler ice rules which very precisely accounts for the difference between the two entropy values [47]. These rules theorised that each oxygen in the water ice lattice was tetrahedrally coordinated and must have two covalent bonds to nearby hydrogen atoms and two hydrogen bonds to more distant hydrogen atoms thus maintaining the familiar H_2O unit [36]. As water ice crystallises with the oxygen atoms at the centres, and the hydrogen atoms at the vertices, of the tetrahedra of a pyrochlore lattice, see figure (13), there are always sixteen possible arrangements of the hydrogen atoms around each oxygen but only six of these satisfy the ice rules. The equivalence of these six states and the fact that the state on any particular tetrahedron does not precisely determine that of

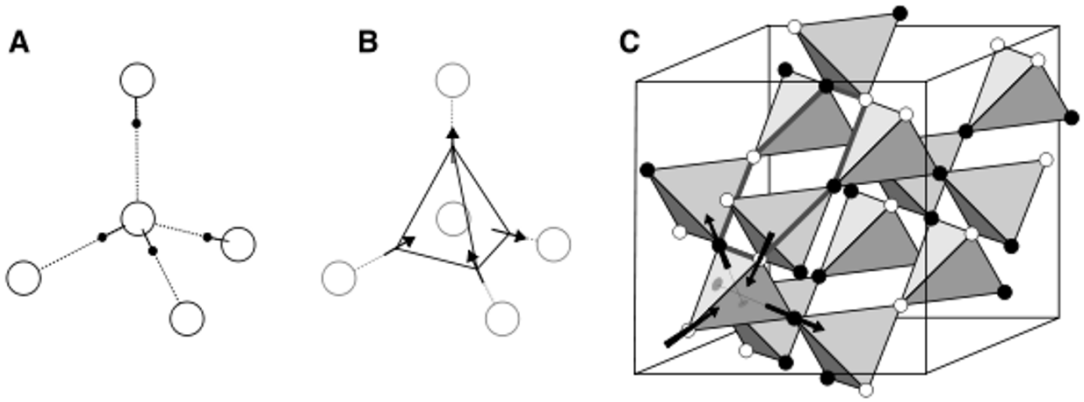

Figure 13: **A**. The local tetrahedral arrangement of oxide ions (open circles) and protons (closed circles) in water ice showing two short bonds (solid lines) and two long bonds (dotted lines) around the central oxide. **B**. Using displacement vectors to represent the same situation as **A** highlights the correspondence between spin direction and hydrogen position in spin ice and water ice. **C**. A pyrochlore lattice of corner-sharing tetrahedra as occupied by the average proton positions in water ice and the magnetic ions in spin ice. Spin directions for one tetrahedron obeying the ice rules are shown with arrows in the lower left corner, all other tetrahedra have spins indicated by open circles (into a downward tetrahedron) and closed circles (into an upward tetrahedron) and the entire lattice obeys the ice rules. Figure taken from reference [32]

its neighbour leads to a macroscopically degenerate groundstate and the zero point entropy which initially caused the difference in entropy measurements. The ice rules constraint on the hydrogen positions was much later verified as correct through neutron scattering observations which are able to provide a microscopic picture of the internal environment of ice [48, 49].

In 1997 Harris *et al.* discovered a magnetic analogue of water ice in the rare-earth titanate $Ho_2Ti_2O_7$ and soon afterwards $Dy_2Ti_2O_7$ was also shown to have the same properties although with hindsight there is evidence of residual entropy in these compounds in much earlier measurements [14, 50]. These materials have a pyrochlore lattice structure with the oxide ions positioned at the centre of each tetrahedron in an exact parallel to the oxygen positions in water ice and the rare earth ions positioned on the vertices of the lattice. The defining feature connecting the two lattices is that the moments of the magnetic rare earth ions are constrained by crystal fields to Ising behaviour along the body centred directions of the tetrahedra. Holmium and Dysprosium share the largest magnetic moment of all naturally occurring ions (10.6 μ_B) and the overall effect of the energetic interactions in the

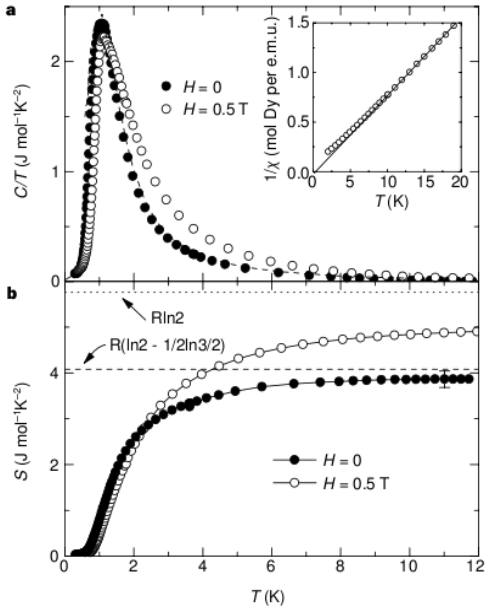

Figure 14: Specific heat, **a**, and Entropy, **b**, measurements taken from the spin ice material $Dy_2Ti_2O_7$. The entropy in zero field tends toward the value calculated by Pauling for water ice, $R(\ln 2 - \frac{1}{2}\ln\frac{3}{2})$, validating the correspondence between the spin ice model and water ice. The application of a field begins to lift the degeneracy of the ground state increasing the entropy toward the limit for a two-state system, $R\ln 2$. The inset shows the high temperature extrapolation of the reciprocal dc susceptibility has a small positive intercept of approximately 0.5K indicating the ferromagnetic nature of the compound. Figure taken from reference [51].

system is ferromagnetic so that two of these large spins point toward the centre and two away from the centre of each tetrahedron in an exact parallel of the two near, two far hydrogen positions in water ice as illustrated in figure (13) taken from Bramwell's review article on spin ice [32]. These materials were therefore described by a model termed spin ice in recognition of the parallels between the two and remain the best experimental examples of the model.

Following the classification of these rare-earth titanates as particularly good examples of the spin ice model Ramirez *et al.* measured the specific heat and entropy of $Dy_2Ti_2O_7$ finding a zero point entropy that agrees very closely with that of water ice and cementing the duality between the models and their experimental realisations, see figure (14) [51]. There have been many discoveries of interest within the area of spin ice physics which we are not able to include in this work including the nature of the true ground state [31, 35], the spin dynamics and susceptibility measurements [52–54] and one area of significant recent interest, the prediction of emergent particles within the lattice that have the characteristics of magnetic

monopoles [55]. This prediction was validated theoretically, [56], and experimentally, [57, 58], and has generated much new work on the spin ice model and materials. Here we attempt only to establish the two standard experimental examples of the spin ice model and demonstrate how closely these materials match the theoretical model.

2.2 Kagome Materials

The kagome lattice has been identified as important for frustrated magnetism but it is difficult to find an accurate expression of it experimentally [59]. Candidate materials are strontium chromium gadolinium oxide (SCGO) [60] and the jarosites [61] although these are not structurally ideal kagome lattices. More recently Kapellasite and Haydeeite, both members of the atacamite family have been proposed as experimental realisations of close to ideal kagome magnets [62].

An alternative approach to an ideal kagome lattice magnet is to isolate the 2D kagome lattice from within a larger 3D lattice. This is readily achievable through the application of a magnetic field along any of the $\langle 1, 1, 1 \rangle$ directions of the pyrochlore lattice of the spin ice materials as originally demonstrated by Matsuhira *et al.* [43]. Deriving from the cubically symmetric pyrochlore lattice the kagome lattice obtained in this way does not suffer from distortions and is a very accurate experimental realisation of the kagome model. As explained in section (1.4.3) the kagome ice state is only stable within a window of temperature and field that allow the spin ice topological constraint to be maintained. Hiroi *et al.* measured the specific heat and entropy of the kagome ice state and were able to map the temperature and field phase space for kagome ice, see figure (15). All of the simulations in this work were carried out at conditions inside the region marked kagome ice on the figure [63].

2.2.1 Neutron Scattering

The spin ice materials $Ho_2Ti_2O_7$ and $Dy_2Ti_2O_7$ are well suited to neutron scattering studies and this technique has been widely utilised from their discovery [14] onward to characterise their structure and behaviour. Neutron scattering is a particularly

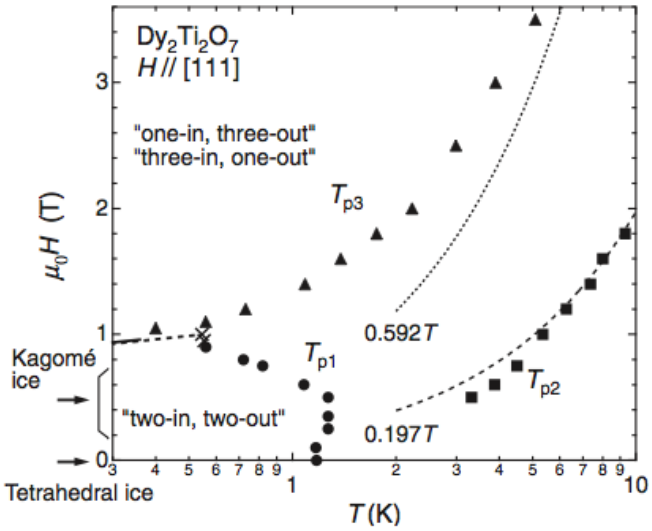

Figure 15: The realisation of kagome ice from the spin ice model is only possible in a restricted temperature and field phase space which has been experimentally determined by Hiroi *et al.* [63]. Outside of the area labelled as kagome ice the divergence free topological constraint on the spin field is not obeyed. Circles, squares and triangles mark the peak positions in the specific heat measurements used to calculate the phase diagram.

powerful method of investigating materials which provides both average properties and spatially resolved information on the spin correlations. A theoretical description of neutron scattering is provided in chapter (3), here we will present only experimental results of interest so that they may be compared with the simulated neutron scattering results in chapter (6).

Bramwell *et al.* confirmed that the rare earth pyrochlore $Ho_2Ti_2O_7$ exhibited the proposed spin ice ground state spin configuration through comparison of the neutron scattering pattern with simulations of the spin ice model using both nearest neighbour and dipolar interactions, see figure (16) [64]. Spin ice has received much attention since its discovery however there is not enough space to review all of the studies here thus, moving straight to the experiments of direct relevance to this work, the pinch point scattering characteristic of the topological constraint in spin ice and hence kagome ice, see figure (17) was recorded by Fennell *et al.* in $Ho_2Ti_2O_7$ and Morris *et al.* in $Dy_2Ti_2O_7$ in the context of investigations into monopoles in these spin ice materials. These works were the first recognition of pinch points in magnetic materials lending further weight to the remarkably accurate description of these materials by the kagome ice model [44, 57, 58].

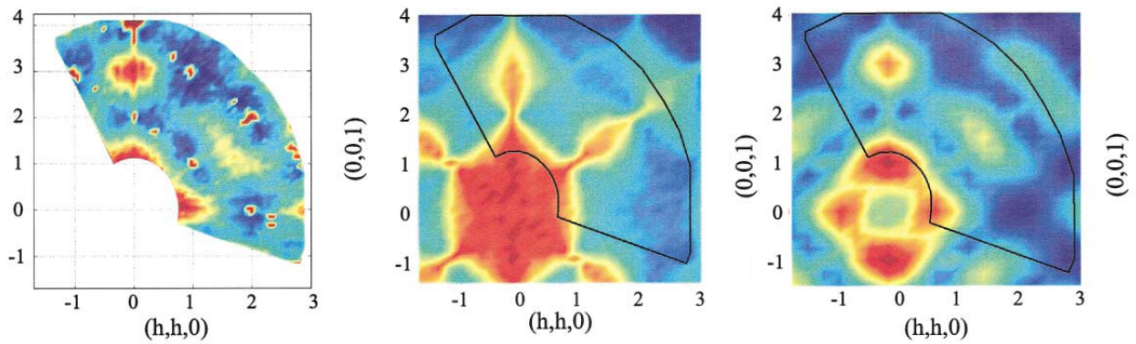

Figure 16: Neutron scattering (left), nearest neighbour simulation (centre) and dipolar simulation (right) of $Ho_2Ti_2O_7$ and the spin ice model confirming the predicted spin ice groundstate, further detail in reference [64].

In order to verify the kagome ice state predicted for spin ice under the influence of a field along a $\langle 1,1,1 \rangle$ direction spin correlations conforming to the kagome ice state were recorded in $Dy_2Ti_2O_7$ under the influence of a magnetic field along the $[1,1,1]$ direction by Tabata *et al*, see figure (18). In this work they showed that as a magnetic field is applied parallel to the $[1,1,1]$ axis of $Dy_2Ti_2O_7$ its spin correlations adjust from the zero field state to those expected for the kagome ice model [65]. This neutron intensity map was also verified by Kadowaki *et al* [66].

Recent work [67] on kagome ice under the influence of an inplane field is shown in figure (19). This experimental scattering pattern was observed whilst the system was being driven from its high temperature disordered state toward the Kasteleyn transition in exactly the same way that the simulations in this work have been performed and these images will be compared with the simulation results in chapter (6). Discussion of the significance of the features in these scattering patterns is also deferred to the results.

2.3 Artificial Square Spin Ice

So far the experimental materials presented have been either naturally very good examples of model magnetic systems or have made use of magnetic fields to isolate experimental realisations of model magnetic systems from within other materials. Recently advances in fabrication at the nano-scale have presented a third way to obtain experimental realisations of magnetic models, to design them rather than

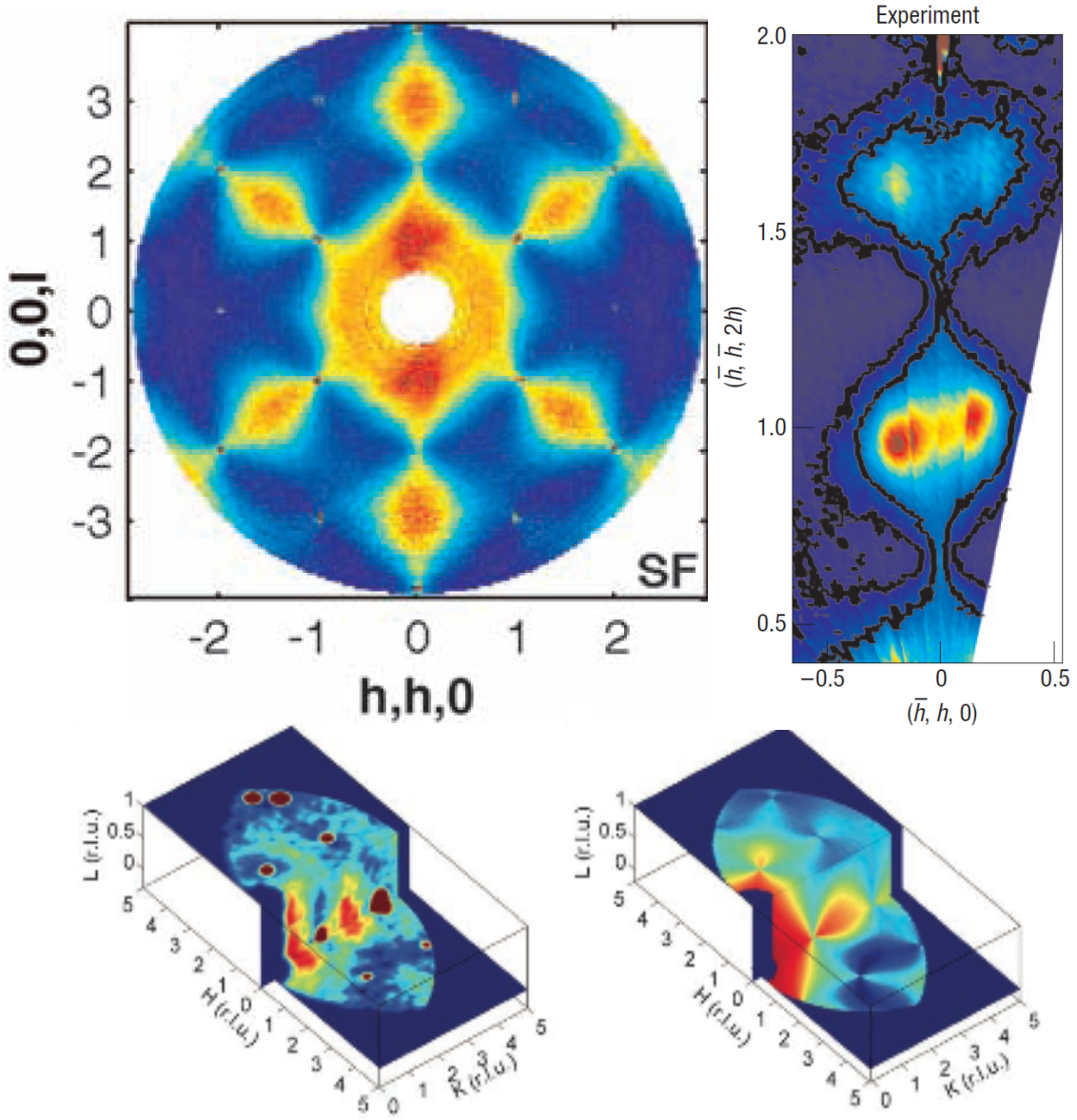

Figure 17: Top left: The polarised component $S^{yy}(\mathbf{Q})$ of the neutron scattering in the spin ice material $Ho_2Ti_2O_7$ displaying pinch points at $(0,0,2),(0,0,4)$ and ten other symmetry related positions [57]. Top right: When a field is applied to $Ho_2Ti_2O_7$ parallel to the $[1,1,1]$ axis it is described by the kagome ice model and also shows pinch points at the positions $\left(\frac{\bar{2}}{3},\frac{\bar{2}}{3},\frac{4}{3}\right)$ and $\left(\frac{\bar{4}}{3},\frac{\bar{4}}{3},\frac{8}{3}\right)$. The vertical axis of this figure corresponds to the bottom left to top right diagonal of figure (18) [44]. Bottom: Neutron scattering data for $Dy_2Ti_2O_7$ at 0.7K (left) and the corresponding correlation function (right) highlighting the 3D nature of the pinch points in reciprocal space [58]. Intensity is measured in arbitrary units where brighter colours indicate higher values.

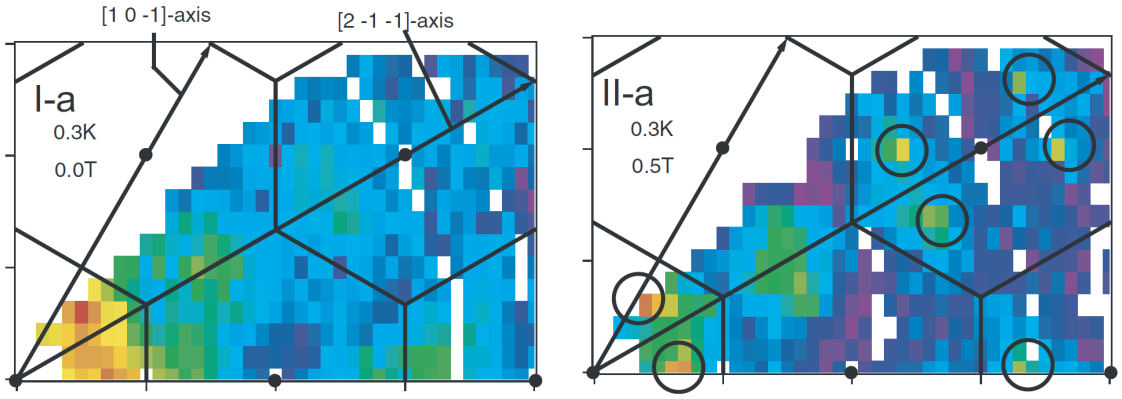

Figure 18: Left: The scattering pattern of $Dy_2Ti_2O_7$ in the plane perpendicular to the $[1, 1, 1]$ direction with no applied field. Right: When a field is applied along the $[1, 1, 1]$ direction of the pyrochlore lattice the scattering pattern changes to reflect the kagome ice state imposed upon the lattice. Black circles show the peaks indicative of the expected correlations in kagome ice. Figure from reference [65].

discover them by bespoke construction in a laboratory. In particular current lithography techniques provide very high resolution (< 10 nm) and enable the fabrication of 2D arrays of magnetic islands each of which is small enough to behave as a single magnetic domain (representing a single spin) at a suitable range of temperatures and fields. These islands can be arranged geometrically to engender frustration into the system and provide nanoscale examples of spin ice and kagome ice [68, 69].

At this point it is again relevant to highlight the difference between a truly 2D kagome ice nanoarray which corresponds to the Wills ice model and does not obey a divergence free constraint for the spins and the kagome ice model in which spins above and below the kagome plane enforce such a restriction. It is unfortunate that nanoarrays with a kagome geometry are referred to as kagome ice as they are fundamentally different from the kagome ice model discussed throughout this work and we attempt to propagate the use of the term Wills ice to describe these systems to prevent further confusion following the model of reference [38]. In this work we focus on kagome ice in the form where it has been isolated from spin ice using a field and within the current context of nanoarrays we restrict our attention to those with a square geometry.

The complete spin ice model has spins interacting with each other in 3D which cannot be replicated on a 2D lattice, however the insight of Wang *et al.* was that by classifying the interactions around each vertex in terms of spins, or islands,

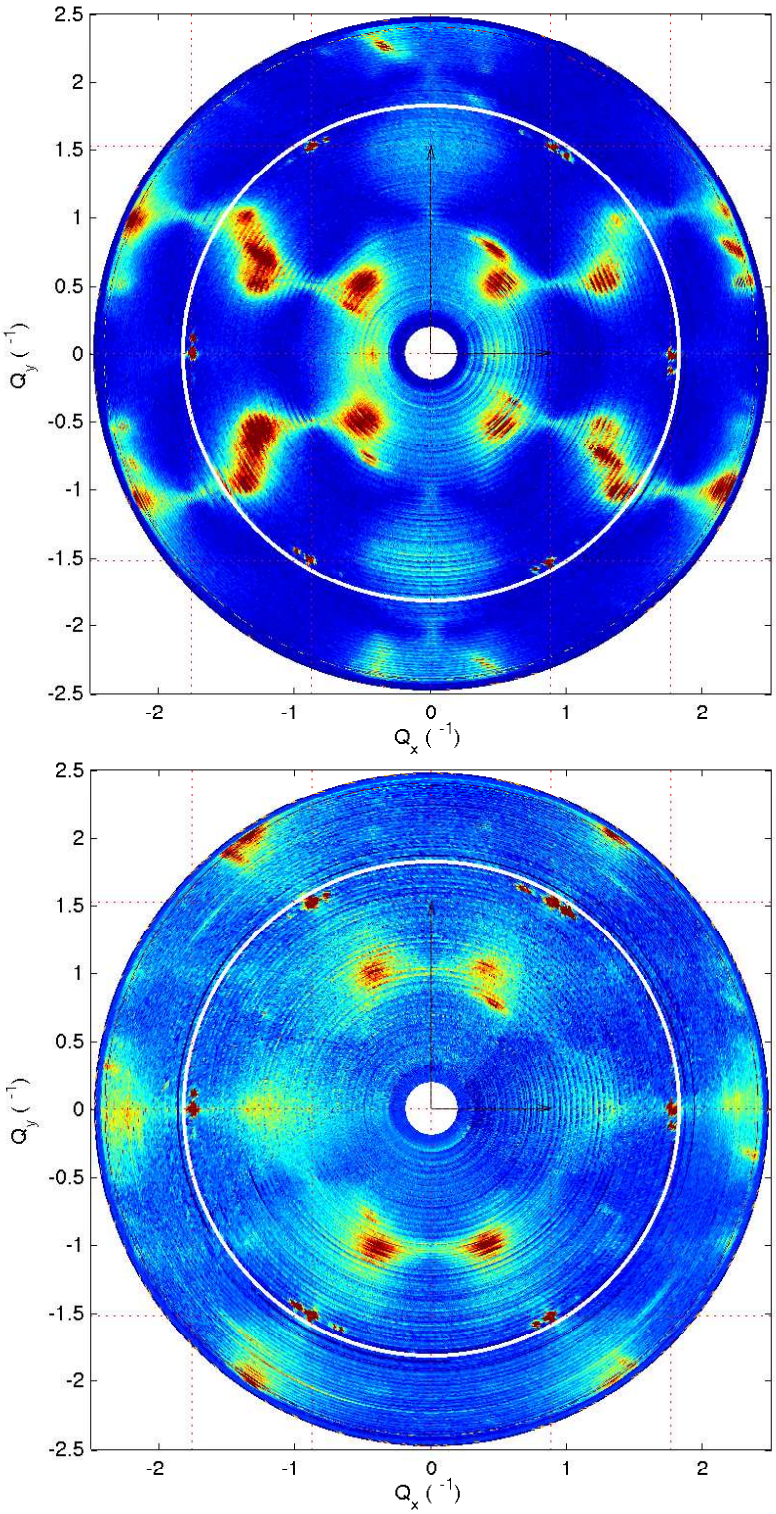

Figure 19: Recent neutron scattering data for $Ho_2Ti_2O_7$ in the kagome ice phase. The scattering vectors $\mathbf{Q}_x$ and $\mathbf{Q}_y$ correspond to the directions $[1, \bar{1}, 0]$ and $[\bar{1}, \bar{1}, 2]$ and higher (arbitrary) intensity regions are denoted by warmer colours. The spin-flip component (top) is equal to the polarisation component in the kagome plane and the non-spin-flip component (bottom) is equal to the polarisation component perpendicular to the kagome plane. Data were obtained in a field of 1 T and at a temperature of 0.07 K with an unknown angle between the field and $[1, 1, 1]$ axis imposed entirely by the shape of the sample [67].

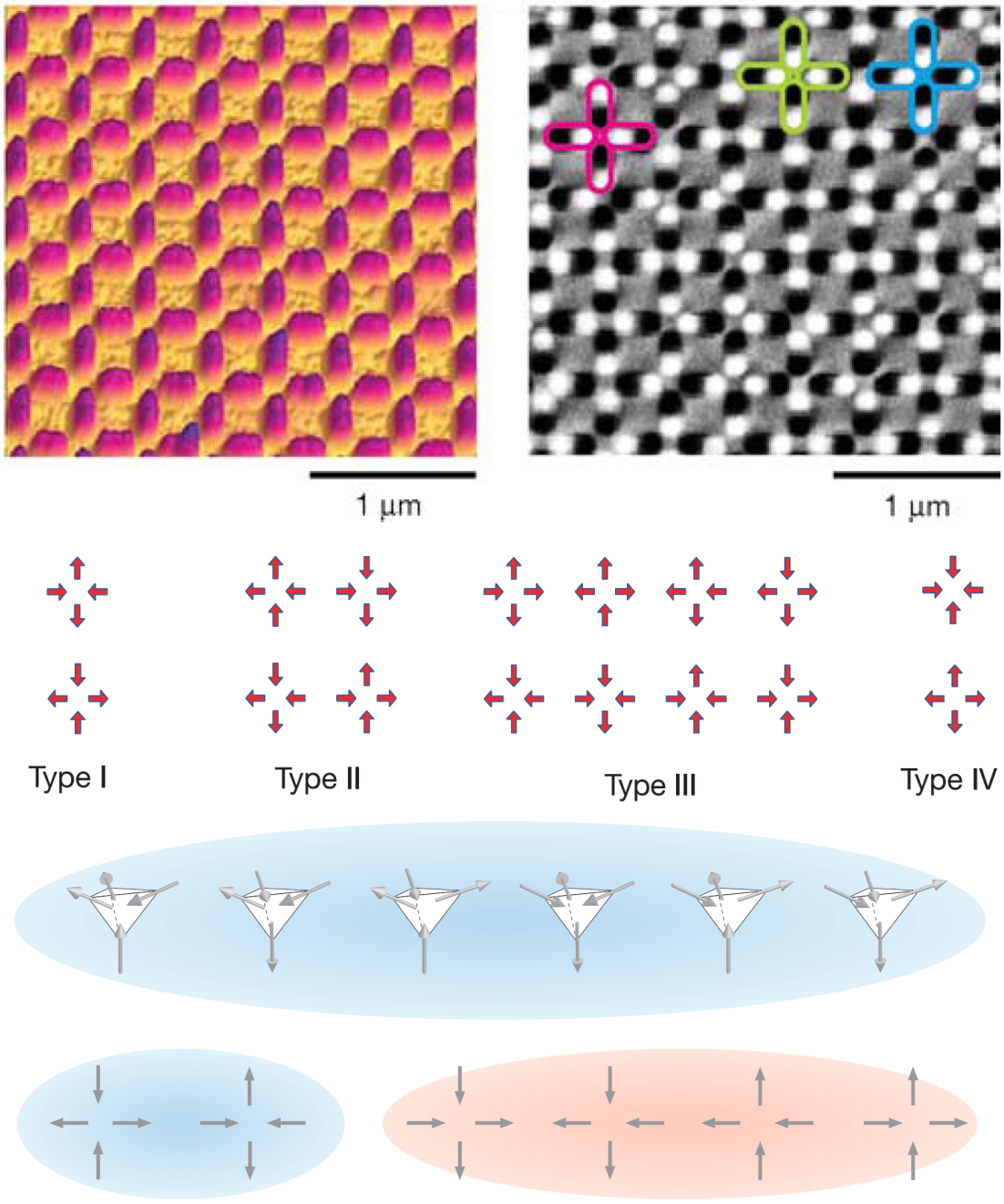

Figure 20: Top left. An AFM image of the first artificial square spin ice nanoarray [68]. Top right. An MFM image of the same array where the bright and dark halves of each island correspond to the magnetic poles. All islands are single domain and therefore a good approximation to individual spins. Centre. The sixteen possible vertex configurations separated by topological type. Type I, II and III are highlighted on the lattice in pink, blue and green respectively. Bottom. The correspondence between the six groundstate spin configurations in 3D spin ice and 2D square ice. Type I vertices and the 3D vertices have no moment (blue) but Type II vertices have a net moment (red). Figure from references [68, 70].

pointing 'in' or 'out' the interactions between ferromagnetic islands arranged on a square lattice were equivalent to the interactions between the spins on the pyrochlore lattice, see figure (20). Analysis of such a square lattice nanoarray showed that just as with spin ice, of the sixteen possible configurations at each vertex (illustrated in figure (20) bottom), those obeying the ice rules are favourable and dominate the statistics of the lattice.

Dipolar interactions between the islands dictate that a staggered arrangement of Type I vertices is in fact a unique ordered groundstate for the lattice although the lack of a thermal energy scale makes this difficult to access. Attempts to thermalise the lattice by rotating the sample in a magnetic field of decreasing strength are able to induce a surplus of ice rule obeying states but are not able to remove all defects from the lattice and Mengotti *et al.* have concluded that for increasingly large systems it will be increasingly hard to reach the dipolar groundstate using this method [71]. In an attempt to overcome this problem Morgan *et al.* have fabricated an array that more closely approaches the true groundstate by thermalising the lattice during the formation process when the very thin islands reduce the energetic barriers to rearrangement however once the nanoarray is complete it has a fixed magnetic configuration and no dynamic behaviour [72].

An appealing feature of magnetic nanoarrays is the ability to examine individual elements or spins directly. This has generated much interest in these systems, particularly after the prediction of magnetic monopoles in true spin ice [55]. In 3D spin ice monopoles are associated with three in, one out or three out, one in vertex defects and as defects of this type are possible in square ice lattices, the Type III vertices, they have been intensely studied as they can be individually interrogated and, for example, tracked across a lattice [73–75]. The lack of a divergence free topological constraint in these nanoarrays removes the emergent gauge field that is one of the requirements of an emergent magnetic monopole and so although 2D magnetic nanoarrays support collective particles of magnetic charge these should not be referred to as magnetic monopoles as they do not strictly obey the definition of a monopole [38].

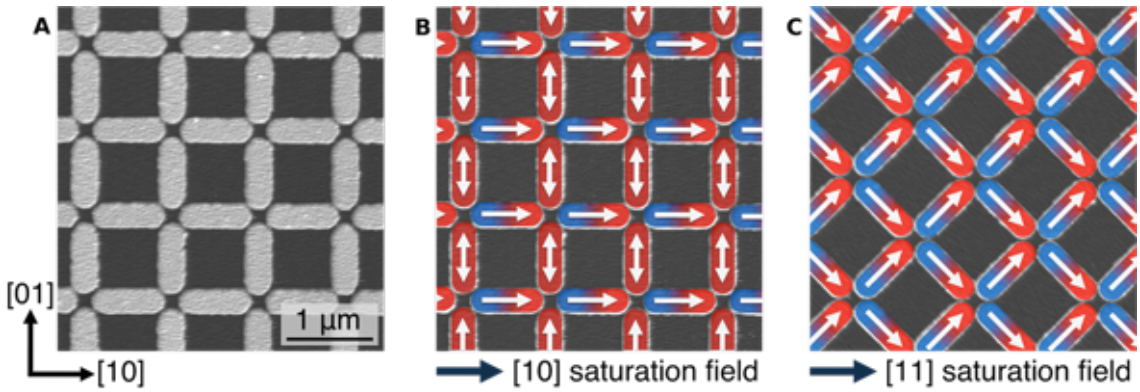

Figure 21: **A**. An AFM image of the square ice nanoarray indicating the scale and major symmetry directions of the lattice. **B**. The remanent magnetic configuration after applying a strong magnetic field in the $[1, 0]$ and **C**. $[1, 1]$ direction. Double headed arrows in **B** indicate islands with indeterminate direction that are uncoupled to the field.

In this work we are particularly interested in the artificial square ice nanoarray fabricated by Kapaklis *et al.* [45] and have written a computational simulation to investigate its properties, the results of which are presented in chapter (7). Experimentally the lack of a thermal energy scale in nanoarrays has been overcome for the first time by fabricating an array using a material with an ordering temperature that is close to room temperature. The lattice is therefore responsive to the effects of temperature and field in a similar way to a classical spin ice and, through experiment and simulation, we show that it possesses true thermal dynamics and exhibits a thermodynamic melting transition.

2.3.1 The Magneto-Optical Kerr Effect

In order to study the response of the nanoarray to temperature and field we have recorded magnetic hysteresis loops at temperatures from 5 K to 300 K in fields applied parallel to the $[1, 0]$ and $[1, 1]$ directions, see figure (21)

The hysteresis loops were recorded using the Magneto-Optical Kerr Effect (MOKE) which describes the manner in which light is altered when it is reflected from a magnetised material [76, 77]. Specifically, the longitudinal set-up was utilised in which light that is linearly polarised is reflected off the surface of the magnetic material in a direction parallel to the magnetisation vector. Interactions between the electromagnetic wave and the magnetic material at the surface change the polarisation

state from linear to elliptical with the change directly proportional to the component of the magnetisation that is parallel to the light beam and the surface. The polarisation is rotated by a very small amount, $< 1°$, but by passing the reflected beam through a polarisation filter perpendicular to the initial polarisation state the intensity of the measured beam is directly effected by the Kerr rotation and this is more readily measured.

These effects initially defied explanation with Kerr stating upon their discovery in 1878 that the observations of Faraday and others along with his discoveries 'must be all included ultimately under one physical theory.' [78]. With the advent of quantum mechanics MOKE is now understood and more recently it was realised that it could be used to sensitively image magnetic domains [79]. The experimental set up makes this method particularly useful for analysing the magnetisation of surfaces and it can be measured over very small surface areas hence MOKE has become widely used in the classification of magnetic thin films and nanoarrays.

3 Introduction to Simulation Techniques

During the 1940s the first general purpose electronic computer was developed at the University of Pennsylvania. The ENIAC (Electronic Numerical Integrator And Computer) was developed for the purpose of calculating artillery firing tables, but the first program run on the computer was the somewhat more complex simulation of a thermonuclear reaction proposed by John von Neumann [80]. From this time onwards computer simulations have played an increasingly large role in scientific research and are now arguably as important as any experimental apparatus. The basis of the first simulation on the ENIAC machine was a statistical approach to treating complex problems known as the Monte Carlo (MC) method [81]. This method has become central to computational simulation and is the method employed in the simulations carried out in this work. The basis of the method is now explained with focus on magnetic models.

The partition function, $\mathcal{Z}$, of a system can be defined as the sum of the Boltzmann weight of each microstate, i, of the system with an associated energy E_i.

$$\mathcal{Z} = \sum_i \exp\left(-\beta E_i\right) \tag{91}$$

Knowledge of this function permits the calculation of thermodynamic quantities such as the magnetisation, specific heat capacity or energy of the system via the fundamental relation between the partition function and the Helmholtz energy, $\mathcal{F}$

$$-\beta\mathcal{F} = \ln \mathcal{Z} \tag{92}$$

For all but the most simple systems however this approach is problematic. Any macroscopic system with N spins where each microstate depends on the arrangement of spins in the system and $N \sim \mathcal{O}\left(10^{23}\right)$ has so many microstates that it is unfeasible to calculate the partition function directly.

This problem highlights an important concept in the relationship between thermodynamics and statistical mechanics. There is no physical way to measure the energy of each microstate of a system, and if there was this information is un-

helpful; quantities measured in experiment such as the magnetisation or energy are equilibrium properties of a system averaged over many microstates which can be quite different to the value of a single microstate. Statistical mechanics provides a framework with which to relate microscopic properties of a system to the required macroscopic properties and is therefore the tool of choice when dealing with numerical simulation where, similarly, individual calculations can only provide microscopic properties but the final results must be equivalent to macroscopic quantities.

The inaccessibility of the partition function does not prevent its use in calculations. A quantum mechanical perspective is a useful starting point here as the idea of an individual microstate of a system is compatible with computational simulation where the system is necessarily in a single state at any single time. It is of course possible to begin classically, and it is helpful to quickly shift to this perspective, but this progression mirrors the connection between a discrete computer simulation and the continuous reality it is attempting to describe. An exact expression for the average of a general quantity $\mathcal{A}$ in quantum mechanical terms is

$$\langle \mathcal{A} \rangle = \frac{1}{\mathcal{Z}} \sum_i \exp\left(-\beta E_i\right) \langle i|\mathcal{A}|i\rangle \tag{93}$$

$$= \sum_i \mathbf{P_i} \mathcal{A}_i \tag{94}$$

where $\langle i|\mathcal{A}|i\rangle$ denotes the expectation value of the operator $\mathcal{A}$ in the state i and $\mathbf{P_i}$ is the Boltzmann distribution [82]. Often the basis set is unknown and it is necessary to introduce a density matrix with non-zero off-diagonal terms to account for this; however, by transforming this equation into its classical analogue it is possible to sidestep this problem.

$$\langle \mathcal{A} \rangle = \frac{\int d\mathbf{p}^N d\mathbf{r}^N \exp\left(-\beta \mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right)\right) \mathcal{A}\left(\mathbf{p}^N, \mathbf{r}^N\right)}{\int d\mathbf{p}^N d\mathbf{r}^N \left(\mathbf{p}^N, \mathbf{r}^N\right) \exp\left(-\beta \mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right)\right)} \tag{95}$$

Sums over states have been replaced by integrals over the momentum and position coordinates of all N particles in the system and the energy of state i is contained within the Hamiltonian $\mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right) = \mathcal{K} + \mathcal{U}$ which expresses the total energy of the system and is equal to the sum of the kinetic and potential energy terms. Both

the Hamiltonian and the observable $\mathcal{A}$ are solely functions of the momentum and position.

The kinetic energy term is quadratic in momentum and therefore it is possible to perform the integration over momenta analytically. For an observable which depends solely on momentum it is generally possible to calculate its average value, but observables usually depend on position and the required integral almost always requires a numerical solution. In the specific case of a magnetic rather than molecular system the spins are angular momenta and terms of order $\mathcal{O}(\mathbf{S}^2)$ are rotational kinetic energy terms. Of these, diagonal terms, $\mathbf{S}_i^2$, are not integrated out of the problem but rather ignored as generally $|\mathbf{S}_i| = \text{constant}$. These magnetic systems are defined as 'classical' because the uncertainty principle is neglected for commutation between $\mathbf{S}_x, \mathbf{S}_y,$ and $\mathbf{S}_z$, which is valid for $|\mathbf{S}| = \text{large}$, but it is interesting to note the integral role of quantum mechanics underlying what is considered as a classical system.

3.1 Metropolis Monte Carlo

Equation (94) indicates that any macroscopic thermodynamic result may be calculated by averaging its value in all microstates of the system. The number of microstates almost always means employing this approach directly is not possible, however it highlights the connection to computational simulation as a model may only exist in a single microstate at any time and similarly the number of possible microstates prevents cycling over all microstates and calculating an average directly. Transforming to a classical reference frame re-casts the sums over states as integrals, and the problem is then to solve the integrals which are still intractable, however the insight of Metropolis *et al.* was that in this situation it is possible to obtain an accurate approximation to an integral by sampling a number of points in its parameter space and averaging over these points [81]. In the limit of an infinite number of points the approximation becomes exact and is equivalent to knowledge of the analytic function, but a reasonably small number of points will give a good approximation without the need to solve the integral. Metropolis *et al.* saw that except

for the function $\mathcal{A}$, the ratio of integrals in equation (95) is equal to the probability distribution around the point $\mathbf{r}^N$ and this distribution is the Boltzmann distribution, which is also clear from equation (94). By generating a number of points in configuration space according to the Boltzmann distribution and then measuring the value of the quantity $\mathcal{A}$ at those points it is possible to calculate an approximation to equation (95) *without* knowledge of the partition function [83]. The Metropolis algorithm is designed to produce a distribution of microstates approaching the Boltzmann distribution such that the system is represented overall in equilibrium and by sampling the system in these states all calculations will contribute usefully to the final result as all the sample points will be representative of the system. Using this approach the MC method avoids the need to include all microstates in calculations with the consequence that the result is not exact. Instead it tends towards the exact result but by restricting the sampled microstates to a much smaller subset of the total the simulation is tractable and able to produce a useful result in the shortest time possible.

The Metropolis algorithm operates by starting the system in a certain microstate then generating a new microstate which is close, in configurational space, to the first and, through application of a simple rule, deciding whether the system should update itself to the new microstate or remain in the original one. Often a microstate with a significant Boltzmann weight under the initial conditions of the simulation is not easily calculated *a priori* and the simulation is begun with the lattice in an easily generated configuration (for example, random or long range ordered) after which there is a period of equilibration during which the algorithm updates the lattice until it is in equilibrium and the microstates are close to a Boltzmann distribution. For microstate i at time t with weight $w_i(t)$,

$$\frac{dw_i(t)}{dt} = \sum_j a_{ji} - a_{ij} \tag{96}$$

where

$$a_{ij} = P_{ij} w_i(t) \tag{97}$$

a_{ij} is the transition rate from state i to j determined by the transition probability P_{ij} and the weight of state i. When the rate of change of w_i is zero the system has reached its equilibrium state. The appropriate number of times to update the lattice in order to reach equilibrium is best determined by taking measurements of a primary thermodynamic quantity which depends on the microstates, such as the magnetisation, as a function of MC steps; at the beginning of the simulation when the lattice is far from equilibrium the value of the quantity will exhibit large fluctuations but once the system has equilibrated it will remain constant except for statistical fluctuations. Monitoring this behaviour during a number of trial simulations provides an approximate number of steps required to equilibrate the lattice however in practice simulations tend to be designed with a significantly larger equilibration period in order to ensure equilibrium has been reached before taking any measurements.

Once the system has reached its equilibrium state it is vital that the updates proposed by the MC algorithm do not destroy that distribution. This can be ensured by stipulating that the average number of accepted trial moves or spin flips that take the system out of an initial state i is exactly equal to the number of accepted spin flips from all other states back into state i. This condition is known as balance and is a necessary and sufficient requirement for any MC simulation. Most MC algorithms, including the Metropolis algorithm, are designed to obey detailed balance; this stronger condition is sufficient but not necessary and stipulates that the number of accepted moves from state i to j is exactly equal to the number of reverse moves from j to i [84]. This clearly satisfies the requirement for equilibrium

that the rate of change of a state i is zero in equation (96).

$$a_{ji} = a_{ij} \tag{98}$$

$$\implies \frac{P_{ji}}{P_{ij}} = \frac{w_i}{w_j} \tag{99}$$

The transition probabilities must also generate an ergodic algorithm so that any state can evolve into any other accessible state in the configuration space. Generally this condition is easily met, but the important parameter is how long it will take the system to evolve between these two states as this informs the length of time it is necessary to perform the simulation and under certain conditions this time may become very large. Near to phase transitions the period of time required for the simulation to remain ergodic almost always increases which is known as critical slowing down. Often there is little that can be done to prevent this problem as it is implicit in the simulation technique however one possible remedy is to use a different method of evolving the system from one microstate to another. This can have a significant impact on critical slowing down and is explained further in the context of the loop algorithm in chapter (5).

Applying the detailed balance condition, equation (98), only to equation (96) would result in systems being unable to alter the distribution of the microstate weights; hence there is a further part of the algorithm which governs the evolution of the microstates and controls the equilibration of the lattice toward the Boltzmann distribution. Once a new microstate has been generated by the algorithm it is not necessarily accepted as an update to the system. There is an acceptance condition for a move from a state i to a state j, $acc(i \to j)$, within the algorithm which may be defined in various ways but in the Metropolis algorithm it is based on the energetic cost of the new configuration, $\Delta E = E(j) - E(i)$.

$$acc(i \to j) = 1, \qquad \Delta E < 0 \tag{100}$$

$$= \exp\left(-\frac{\Delta E}{k_B T}\right) \geq rand, \qquad \Delta E \geq 0 \tag{101}$$

where $rand$ is a random number on the interval $[0, 1]$. This means that even if the

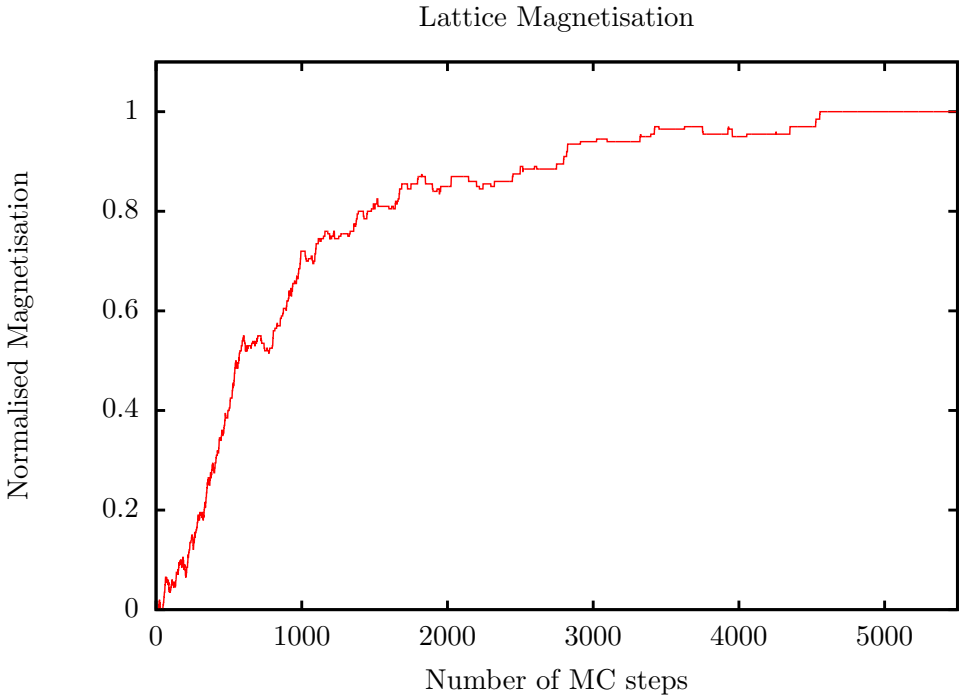

Figure 22: Normalised magnetisation of the kagome lattice as a function of single spin flip MC updates in a field. This data was calculated for the kagome ice model described in section (1.4.3) using the nearest neighbour exchange Hamiltonian, equation (55). The lattice was initialised in a random configuration so that the average magnetisation of the lattice was approximately zero at the beginning of the simulation. The magnitude of the magnetisation is proportional to the weight of the microstates and so its variation reflects their evolution toward the Boltzmann distribution. The magnetisation shows an overall trend toward its saturation value but over a small scale its progression is random. In this case after approximately 4600 MC steps the magnetisation has reached its saturation value and further variations in the magnetisation are entirely due to statistical fluctuations which are too small to see on the scale of this figure. Once a constant value has been reached for a quantity that depends on the weight of the microstates of the system the model has reached its equilibrium distribution.

simulation is started with the system in a very unfavourable state it will quickly evolve to a distribution of states with significant Boltzmann weights, see figure (22).

The use of random numbers in MC modelling is crucial as it allows the system to avoid local minima in the energy landscape. It is also the key to enable MC simulations to incorporate an element of reality in the sense that an entirely deterministic simulation uses a computer to quickly calculate something that could have been calculated, albeit very slowly, given knowledge of the state of the model before the simulation started, whereas MC results cannot be predicted exactly because the input parameters do not completely specify the behaviour of the simulation. As part of

the procedure for updating the system depends on the random numbers, the way in which these are generated is important. The numbers need not be entirely random, rather Metropolis *et al.* indicate they should be a series of uncorrelated numbers uniformly distributed on the interval $[0, 1]$. In the earliest Monte Carlo simulations the pseudo-random number generator was von Neumann's middle square algorithm [85], however this was not an ideal generator and current simulations often use generators with very long repeat periods that pass strict statistical tests regarding the correlation between the numbers. Von Neumann used a pseudo-random number generator as it was faster than true random number generators based on hardware, and, as a pseudo-random sequence is repeatable given the same initial parameters, it is possible to repeat experiments, and therefore isolate errors in simulations.

Calculating each microscopic result is generally a simple mathematical operation. The following equations specify, per spin $\mathbf{S}_i$, some standard the definitions used in this work for the thermodynamic quantities of interest, the exchange energy (E), magnetisation ($\mathbf{M}$), magnetic susceptibility (χ), and the specific heat (C).[2]

$$E = \frac{1}{N}\left\langle -J \sum_{i,j(n.n.)}^{N} \mathbf{S}_i \cdot \mathbf{S}_j - \mathbf{H} \cdot \sum_{i}^{N} \mathbf{S}_i \right\rangle \tag{102}$$

$$\mathbf{M} = \frac{1}{N}\left\langle \sum_{i}^{N} \mathbf{S}_i \right\rangle \tag{103}$$

$$\chi = \frac{N}{T}\left(\left\langle \mathbf{M}^2 \right\rangle - \langle M \rangle^2\right) \tag{104}$$

$$C = \frac{N}{T^2}\left(\left\langle E^2 \right\rangle - \langle E \rangle^2\right) \tag{105}$$

where

$$N = \text{The number of spins on the lattice.}$$

$$\sum_{i,j(n.n.)}^{N} = \text{Sum over all nearest neighbours } j \text{ to spin } i \text{ for all } i.$$

[2] In physical systems the magnetisation is typically specified as the extensive magnetic moment per unit volume, $M = \frac{N}{V}m$, however this does not have a well defined meaning when using a lattice with periodic boundaries to simulate a material. In this work we specify quantities as 'per spin' as this retains a precise meaning.

The magnetisation and the energy are primary quantities as they can be calculated directly from the model, whereas their fluctuations such as the susceptibility and specific heat require average values of these quantities and can therefore only be calculated after a series of microstates have been sampled. The magnetisation is a vector quantity but often, in the calculation of the susceptibility for example, only the magnitude is required which is calculated as the Euclidean norm,

$$M = \sqrt{\mathbf{M}_x^2 + \mathbf{M}_y^2} \tag{106}$$

During simulations the lattice is initialised in an easily generated configuration, generally long-range order if the initial simulation conditions are low temperature or high field, or random order if the initial conditions are high temperature or low field. The lattice is then equilibrated by calling the lattice update algorithm a predetermined number of times, as explained above, after which the system will be in a microstate with a significant Boltzmann weight and the measurement sampling is started. The measurement routine is called to take measurements of the system in its current microstate, after which the update algorithm is called one or more times to allow the lattice to evolve to a new microstate. A larger number of measurements produces better accuracy and smaller statistical noise in the final result. Once the required number of measurements has been reached the primary quantities are recorded as the arithmetic mean of their measurements which also allows the fluctuations in those quantities to be evaluated. The external parameters are then updated and the cycle starts again with a new period of equilibration. Finally after the thermodynamic quantities have been evaluated at each value of the external parameters required by the simulation the quantities are averaged by the number of measurements and number of spins in the lattice to produce intensive, or per spin, values which are output as a datafile.

3.2 Magnetic Neutron Scattering

Thermodynamic quantities provide information about the macroscopic properties of a frustrated magnetic material and can indicate transitions between phases, but

they reflect only equilibrium averages of the material and do not reveal anything of its internal structure. Scattering a beam of radiation, in this case neutrons, from the sample under suitable conditions, to be determined in the following discussion, probes the spin to spin correlations within the material and so provides a more detailed description. The power of magnetic neutron scattering is that it can clearly distinguish between correlations of differing symmetries, which typically correspond to different phases, and then from the correlations elucidate the internal spin structure which reflects the nature of the interactions between the spins and is particularly relevant in frustrated magnetic systems.

A feature which makes neutrons valuable for experiment is their interaction with nuclear and magnetic, but not electrical forces. X-rays, which interact with the electrons in an atom, are scattered, to a first approximation, increasingly strongly by elements with an increasing atomic number so that they are insensitive to scattering from light elements, and it is very difficult to distinguish between neighbouring elements. Neutrons are scattered by the nuclei of atoms, or their magnetic moments, thus the scattering intensity is not affected by the electronic structure of the atomic core. This makes neutron scattering a useful tool to investigate both light atoms, and some neighbouring atoms which may be almost indistinguishable to X-rays, but may have significantly different scattering properties for neutrons, see figure (23).

Numerical calculation of neutron scattering intensity maps is well suited to computational simulation as the procedure is simple but requires intensive numeric processing. The theory of magnetic neutron diffraction is now presented, followed by a description of the implementation of this procedure in computational simulation. The presentation is necessarily brief for a subject that has monographs devoted entirely to its description and is based on reference [87].

A neutron scattering experiment is initiated by directing a beam of neutrons with wavevector $\mathbf{k}$ onto a periodic structure, often this is the atomic lattice that defines the material, but in the context of magnetic materials this is the periodic structure formed by the magnetic moments within the material which may not have the same periodicity as the structural lattice. The neutrons that are scattered by the

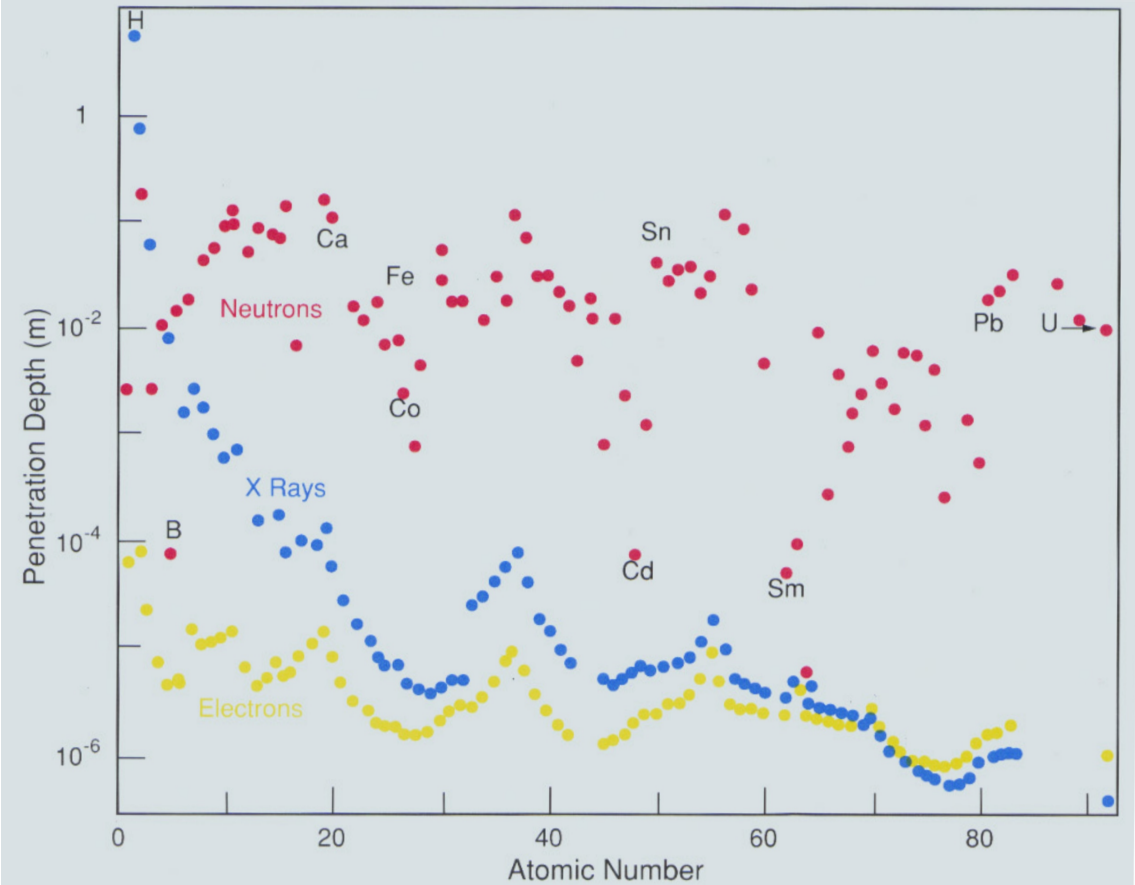

Figure 23: Penetration depth into pure elemental materials of thermal neutrons with a wavelength of 14 nm, X-rays and electrons until the beam intensity has been reduced by a factor of e^{-1}. X-rays are scattered by the electrons in an atom and so, to a first approximation, penetrate to a depth which decreases with increasing atomic number. Neutrons are scattered by the nuclei, or magnetic moments, hence their penetration depth is not related to atomic number. Figure from reference [86].

material are then measured and the difference between their properties before and afterwards provide information about the internal environment of the material. The appropriate 'language' for this description is that of crystal structure and reciprocal space, due to Gibbs [88], which is recapped first.

Any point, $\mathbf{r}$, on a periodic lattice of magnetic moments may be defined in terms of the lattice basis vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$,

$$\mathbf{r} = m\mathbf{a} + n\mathbf{b} + p\mathbf{c} \tag{107}$$

where m, n and p are integers. There is also a reciprocal lattice to the real space, or direct, magnetic lattice with reciprocal lattice vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ which are inversely related to the basis vectors of the direct lattice and any point on the reciprocal magnetic lattice, $\mathbf{R}$, may be defined,

$$\mathbf{R} = h\mathbf{A} + k\mathbf{B} + l\mathbf{C} \tag{108}$$

where h, k and l are integers.

The incident beam of neutrons in a scattering experiment is a plane wave described by

$$\psi = \exp(i\mathbf{k} \cdot \mathbf{x}) \tag{109}$$

with $\mathbf{x}$ defined as the direction of the incident beam. In the simplest case the neutrons are scattered from objects with very short ranged potentials which can be considered as located at a single point, for example the nuclei of atoms. The outcome of the interaction is a spherically symmetric circular wave originating from each scattering centre,

$$\psi' = -\frac{b_{\mathbf{r}}}{r} \exp(i\mathbf{k}' \cdot \mathbf{r}) \tag{110}$$

where $\mathbf{r}$ is the position vector of the scattering centre of magnitude r, $\mathbf{k}'$ is the new wavevector and $b_{\mathbf{r}}$ is the scattering length which describes how strongly the incoming neutrons are scattered by the object at $\mathbf{r}$. The spherical waves interfere with each other as they propagate so that generally a spherical wave from any

nucleus is cancelled out by the destructive interference of all other spherical waves, but in certain cases they can interfere constructively to produce what appears to be a reflection of the beam with wavevector $\mathbf{k}'$ at an angle 2θ to $\mathbf{k}$ so that the scattering vector can be defined,

$$|\mathbf{Q}| = |\mathbf{k} - \mathbf{k}'| = \frac{4\pi \sin\theta}{\lambda} \tag{111}$$

When neutrons are scattered by longer range magnetic dipole interactions with the spins in atoms rather than their nuclei the resulting scattered wave has a more complex form than equation (110) illustrated by the experimentally measured quantity, equation (118). This takes account of both the longer ranged interaction and the spatially extended scattering center presented by the spins of an atom.

The measurable quantities in neutron scattering experiments are the momenta of the incoming and outgoing waves (or neutrons) and the variable parameters for experimenters are therefore the energy and direction of the incoming neutrons. In this derivation only elastic scattering is considered in which the magnitude of the incoming and outgoing wavevectors is the same and there is no energy transferred to the material. The timescale in which a neutron interacts with a scattering centre is very small and it can be assumed that it does not affect it during this period. This is known as the static approximation and leaves one experimental variable, the scattering vector $\mathbf{Q}$.

In order for the incoming and outgoing wavevectors to have the same magnitude the scattering vector must be perpendicular to the planes of scattering centres. A further condition for diffraction of the neutrons is that the distance between the consecutive planes must satisfy the condition $\mathbf{Q} \cdot \mathbf{d} = 2\pi n$ where n is an integer. Combining this with equation (111) produces Bragg's law [89],

$$2d \sin\theta = n\lambda \tag{112}$$

Bragg's law states that there will only be coherent scattered radiation when the wavelength and the scattering angle satisfy a particular relationship. This relationship can also be expressed in terms of the scattering vector as the condition for

scattering is [2],

$$\mathbf{k}' = \mathbf{k} + \mathbf{Q} \tag{113}$$

$$\implies k'^2 = k^2 + 2\mathbf{k}\cdot\mathbf{Q} + Q^2 \tag{114}$$

elastic scattering enforces

$$k'^2 = k^2 \tag{115}$$

$$\implies 2\mathbf{k}\cdot\mathbf{Q} + Q^2 = 0 \tag{116}$$

The incoming flux of neutrons is scattered by the magnetic field of the sample, $\mathbf{B}(\mathbf{r})$, which is due to the spins and variable with position. This suggests the central result of magnetic neutron scattering which is that the quantity measured during a static scattering experiment is proportional to the spatial Fourier transform of the two spin correlation function known as the static scattering function,

$$S^{\alpha\beta}(\mathbf{Q}) = \sum_{\mathbf{R}} \langle \mathbf{S}_0^{\alpha} \mathbf{S}_{\mathbf{R}}^{\beta} \rangle \exp\left(i\mathbf{Q}\cdot\mathbf{R}\right) \tag{117}$$

where the angled brackets indicate a thermal average and α and β are cartesian components of the spins.

For the large majority of values of $\mathbf{Q}$ after completing the sum over lattice positions the phase factors cancel which is equivalent to the destructive interference of the scattered spherical waves mentioned previously, however when $\mathbf{Q}$ is equal to a reciprocal lattice vector, $\mathbf{Q}\cdot\mathbf{R} = 2\pi n$ where n is an integer, $S^{\alpha\beta}(\mathbf{Q})$ obtains its maximum value.

During a scattering experiment the ideal quantity to measure is the scattering cross section, σ, which is the total number of neutrons scattered by the material relative to the incident flux of neutrons, but an experimental detector can only measure the scattered neutrons in a small region of solid angle, $\partial\Omega$, at any time hence the differential cross section is defined as a measure of the number of neutrons scattered into a small solid angle around a particular scattering angle and it is this

quantity which is recorded experimentally.

$$\frac{\partial\sigma}{\partial\Omega} = r_0^2 \frac{|\mathbf{k}|}{|\mathbf{k}'|} f(\mathbf{Q})^2 \sum_{\alpha\beta} (\delta_{\alpha\beta} - \hat{Q}_\alpha \hat{Q}_\beta) S^{\alpha\beta}(\mathbf{Q}) \tag{118}$$

This quantity differs importantly from the static scattering function. The summation term, $\sum_{\alpha\beta}(\delta_{\alpha\beta} - \hat{Q}_\alpha\hat{Q}_\beta)$, selects only the components of each spin that are perpendicular to the scattering vector as it is only these that interact with the neutrons.

The static scattering function is the spatial Fourier transform of the two-spin correlation function but may also be expressed in terms of the wavevector dependant magnetisation of the sample, $\mathbf{M}(\mathbf{Q})$.

$$S^{\alpha\beta}(\mathbf{Q}) = \frac{1}{N}\langle \mathbf{M}^\alpha(\mathbf{Q}) \cdot \mathbf{M}^\beta(-\mathbf{Q})\rangle \tag{119}$$

where

$$\mathbf{M}(\mathbf{Q}) = \sum_{\mathbf{r}} \mathbf{S}_\mathbf{r} \exp\left(i\mathbf{Q}\cdot\mathbf{r}\right) \tag{120}$$

If the magnetic moments are fully ordered within the sample then the scattering will be entirely composed of Bragg peaks and the differential cross section can be expressed,

$$\frac{\partial\sigma}{\partial\Omega} = r_0^2 \frac{|\mathbf{k}|}{|\mathbf{k}'|} f(\mathbf{Q})^2 \left| \sum_\rho \langle \mathbf{S}_\perp(\rho)\rangle \exp\left(i\mathbf{Q}\cdot\rho\right) \right|^2 \delta(\mathbf{Q}-\mathbf{R}) \tag{121}$$

$$= r_0^2 \frac{|\mathbf{k}|}{|\mathbf{k}'|} f(\mathbf{Q})^2 \left|\mathbf{M}_\perp(\mathbf{Q})\right|^2 \delta(\mathbf{Q}-\mathbf{R}) \tag{122}$$

where the selection of the perpendicular components of the spins is now indicated by the $\perp$ symbol and the summation is only required over the vectors ρ describing the atomic positions in a single unit cell, however if the there is some disorder in the lattice it is necessary to take the summation over all spin positions of the lattice as in equation (120). $|\mathbf{M}_\perp(\mathbf{Q})|^2$ is the called the magnetic structure factor and is the quantity simulated numerically. By incrementally altering the scattering vector

Q the simulation is able to map out the neutron scattering response over a region of reciprocal space. Figure (24) shows the result of this process on an equilateral triangle of three spins and a long-range ordered kagome lattice. The scattering from the triangle shows variation in intensity as a function of the scattering vector but as there is no long range order to diffract the incident wave in a coherent way there are no Bragg peaks. The long-range ordered kagome lattice shows only Bragg scattering as bright spots on the scattering vectors that coincide with lattice vectors of the reciprocal magnetic lattice. In the limit of an infinitely large system these peaks tend to delta functions.

During a real neutron scattering experiment the differential cross section is measured over a period of time which is long compared to spin fluctuations in the sample hence it represents the scattering from an equilibrium spin configuration. Simulations represent a similar equilibrium spin configuration by performing the scattering calculation over all values of the scattering vector on a series of uncorrelated spin configurations generated by a separate MC simulation and then averaging the result.

During a neutron scattering simulation a series of files containing the coordinates and states of all spins in the lattice (which are generated during a separate simulation) are processed in order to obtain a good statistical average corresponding to an equilibrium spin distribution. Each file contains one lattice configuration and the program is required to loop over all the specified points in reciprocal space and loop over all the spins positions on the lattice at every point, therefore the calculation of scattering maps is computationally expensive but valuable as the information provided is detailed and not easily otherwise available.

3.2.1 Consequences of Dipolar Correlations

The utility of neutron scattering is immediately apparent when investigating spin ice materials. In the thermodynamic limit the local ice rule on each tetrahedron of the spin ice lattice (two spins in, two spins out) produces a divergence free, topological constraint on the magnetic moment field,

$$\nabla \cdot \mathbf{M}(\mathbf{r}) = 0 \tag{123}$$

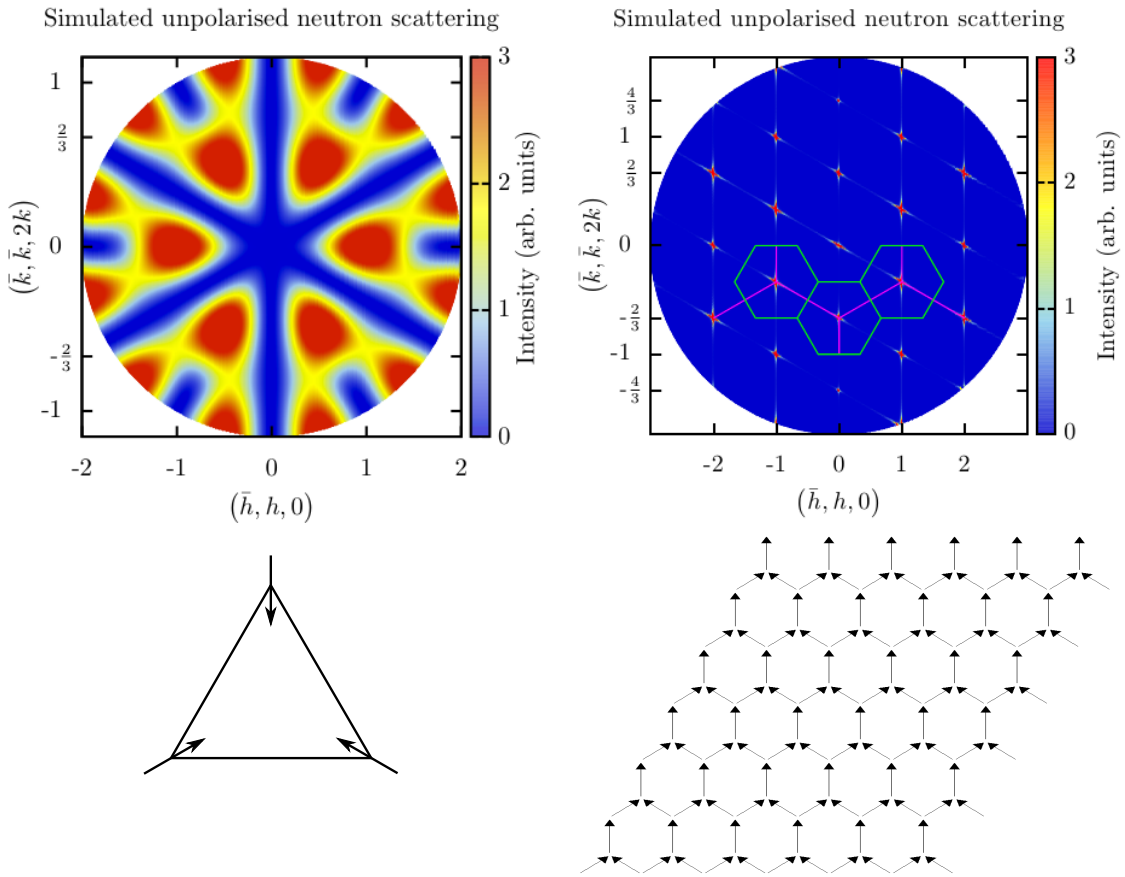

Figure 24: Neutron scattering maps (above) and the spin configuration that generated them (below). Left: The scattering from three spins on an equilateral triangle. Averaged over the three allowed configurations on an up triangle the spins all point in. The pattern does not show Bragg peaks as there is no long-range spin order to diffract the incident neutrons but already shows the six-fold symmetry and central region of low intensity surrounding by six bright regions which is observable in the complete unpolarised scattering pattern for kagome ice, figure (39). Right: Bragg scattering from an ordered kagome lattice. Scattering only occurs at the values of the scattering vector that coincide with the reciprocal magnetic lattice vectors and is concentrated into very sharp peaks. The symmetry of the pattern still reflects the underlying triangular lattice motif.

This solenoidal condition restricts the total magnetic flux entering each tetrahedron to be equal to the total flux leaving it. A consequence of this is that the internal energy of each tetrahedron in the lattice (or triangle in the kagome ice lattice) is equivalent (within the nearest neighbour model) and may be disregarded leaving the balance between Zeeman energy and entropy as the driving energetic force in the system. A more complex consequence is that although this topological constraint is enforced entirely through local energetic considerations it produces the appearance of a system containing long ranged dipolar interactions. Youngblood, Axe and McCoy first noted that correlation functions are linked to charge interactions and can provide information about their form [90]. In that reference they remind the reader that typically the critical scattering around a Bragg peak has a Lorentzian form and it is shown that the correlation function governing such scattering behaves as a monopolar field which is rotationally invariant. In the case of copper formate tetrahydrate which is a 2D system with ice rules similar but not identical to kagome ice, the correlation function is shown to take the form of a dipolar field which is not rotationally invariant and leads to unusual scattering peaks. Using the correspondence between the correlation function and its associated charge interaction it is possible to investigate the manner in which the local topological constraint leads to neutron scattering patterns with the characteristics of those produced by a system with dipolar interactions.

The analysis is straightforward in 2D which is appropriate to the kagome lattice but it remains valid in 3D. Consider an orthogonal coordinate space with z perpendicular to the scattering vector and x and y in the plane of the scattering vector, then examine the Fourier transform of the constraint in order to examine its consequences on the scattering patterns,

$$\nabla \cdot \mathbf{M}(\mathbf{r}) = 0 \tag{124}$$

$$\implies \mathbf{Q} \cdot \mathbf{M}(\mathbf{Q}) = 0 \tag{125}$$

$$\implies Q_x M_x(\mathbf{Q}) + Q_y M_y(\mathbf{Q}) = 0 \tag{126}$$

implying that the system must be completely described by Fourier components for

which the magnetic moment is perpendicular to its wavevector and the perpendicular components of the magnetisation are not independent of each other. A simple phenomenological form for the Gibbs energy of the system in the presence of a magnetic field parallel to the x-axis can be written as a sum of wavevector magnetisation components and a field interaction term,

$$\mathcal{G} = \sum_{\mathbf{Q}} \frac{1}{2\chi_0} \left(M_x^2(\mathbf{Q}) + M_y^2(\mathbf{Q}) \right) - M_x(\mathbf{Q}) H_x(\mathbf{Q}) \tag{127}$$

where the component of the susceptibility that is independent of the wavevector, χ_0, is given by the Curie relation. M_x or M_y can be eliminated from this equation using equation (126) after which minimising with respect to the same magnetisation component provides the susceptibilities,

$$\chi^{xx} = \chi^{yy} = \chi_0 \frac{\mathbf{Q}_y^2}{\mathbf{Q}_x^2 + \mathbf{Q}_y^2} \tag{128}$$

$$\chi^{xy} = -\chi_0 \frac{\mathbf{Q}_x \mathbf{Q}_y}{\mathbf{Q}_x^2 + \mathbf{Q}_y^2} \tag{129}$$

The Ornstein-Zernike form of the scattering function shows that it is directly proportional to the susceptibility thus these equations illustrate the behaviour of the scattering [87]. There is a singularity in the function as the wavevector approaches zero but it has an angular dependence. If the origin is approached with $\mathbf{Q}_y = 0$ then the susceptibility is zero, but if the origin is approached with $\mathbf{Q}_x = 0$ then the susceptibility is finite if the directions are the same and zero if they are different. When examined over all angles of the wavevector this produces a scattering pattern with a characteristic bow-tie shape known as a pinch point illustrated in figure (25).

Scattering patterns containing pinch points such as those presented in chapter (6) indicate long ranged correlations due to the reciprocal relation between momentum and real space. In order to produce scattering which is constricted to a singularity in the centre of the pinch point and hence extends over a vanishingly small reciprocal length the real space lattice must contain correlations extended to an infinitely large distance. This requirement is never perfectly fulfilled as rare violations of the topological constraint introduce a long length scale into the system which creates

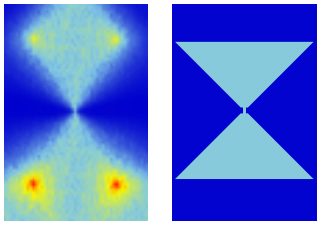

Figure 25: Left: A magnified region of the neutron scattering map of kagome ice simulated in this work showing a pinch point with its characteristic bow tie shape. Right: Schematic highlighting the shape. In an idealised pinch point the scattering reduces to an infinitely sharp pinch point at the centre however rare violations of the topological constraint produce a long length scale within the system at which the correlations break down producing a small central ridge of scattering as shown.

a small ridge of scattering rather than a complete singularity in the centre of the pinch point. This can be accounted for by adding a term to the phenomenological free energy due to ice rule violations which modifies the susceptibility to give for example,

$$\chi^{xx} = \chi_0 \frac{\mathbf{Q}_y^2 + l_d^{-2}}{\mathbf{Q}_x^2 + \mathbf{Q}_y^2 + l_d^{-2}} \tag{130}$$

where l_d is a length scale associated with the distance between ice rule violations which relaxes the condition for a singularity and broadens the central point as shown in figure (25) and figure (3) of [91]. This behaviour has been verified experimentally using neutron scattering on a spin ice material where the observed pinch points were well described by adding an ice rules defect length to the idealised pinch point description in a similar way to that shown above [57].

4 The Kasteleyn Transition

A large proportion of the work presented in this chapter is a review, analysis and consolidation of previous work on the Kasteleyn transition which is necessary firstly to provide context for the simulations carried out in this work but also as this highly complicated transition benefits from the presentation of its salient points in a single location. It is also worthwhile presenting a justification for the effort invested here in simulating an exactly solvable model. Whilst the model has analytical solutions these do not cover the entire phase space and where they do exist their complexity easily leads to errors as we shall show when discussing the neutron scattering results presented in chapter (6) compared with those in the literature calculated using analytic methods. When the applied field subtends a finite angle to the $[\bar{1}, \bar{1}, 2]$ direction the results of the kagome ice, and the more general Kasteleyn model, are not available without using simulation and the work we present in this area is new. As discussed in chapter (6) this can necessitate a reappraisal of the theoretical analysis of the transition and the testing of hypothesis is suitable justification for performing it.

4.1 Lattice Dimer Coverings

After the success of the Ising square lattice and other similar Ising problems the statistical mechanics of lattices received further interest through the closely related problem of lattice dimer coverings [92]. A hard core dimer is any line connecting two neighbouring lattice points which may be placed on the lattice according to a specific rule, for example on the square lattice one possible rule is that a dimer must lie along the bond between two lattice sites, and may not overlap with its neighbours so that any lattice point may only be covered by the endpoint of a single dimer [93]. The statistical properties of the combinatorial problem of placing the dimers on the lattice can be calculated, and from this information it is possible to extract the thermodynamic properties of the lattice. Just as with Ising spins, it is often possible to form a long-range ordered arrangement at low temperatures or high fields that can be transformed into a random dimer covering by changing the

external parameters of the system such as the temperature or field, and the system will show evidence of a phase transition between the two states.

There is a general class of systems in which hard core dimers may be placed onto bipartite lattices and the restrictions controlling the placement may be expressed as a topological constraint [94]. In a subset of this group the lowest energy excitations of the lattice above its long range ordered state are also topological objects, typically strings that span the lattice, and in this case it is possible to observe an unusual form of phase transition where there are no fluctuations in the internal energy of the system but nevertheless there is a singular temperature at which an external field can induce a phase transition. It is this particular case that is relevant to the work presented in this chapter.

Kasteleyn studied the ordering of dimers on the bonds of the honeycomb lattice, see figure (26), and noted that the application of a local chemical potential for the dimers, μ_i, along one of the $i = 1, 2, 3$ possible dimer orientations produced a singular ordering transition for finite fugacity, $z_i = \exp(\beta\mu_i)$, now known as the Kasteleyn phase transition [42]. Hydrocarbon chains in lipid bilayers show an ordering transition with an asymmetric character which Nagle classified as a Kasteleyn transition [95] and it has also been used to explain phenomena as diverse as (anti)ferroelectric ordering transitions [96, 97], polymer melting transitions [98], and models of domain walls [99]. Moessner and Sondhi revived interest in this transition by observing that the kagome ice model satisfies the criteria for a Kasteleyn transition [33], see figure (27), which provided the starting point for a large proportion of the work in this thesis. Recently it has also been observed in perhaps one of its purest examples in 3D in the complete spin ice model [100].

4.2 Topological Sectors

The spin ice model has a local constraint that two spins must point in toward the centre and two must point outward on each tetrahedron. In this case it is straightforward to see the link between this ice rule and the topological constraint which is solenoidal in nature; the total magnetic field entering a tetrahedron is equal

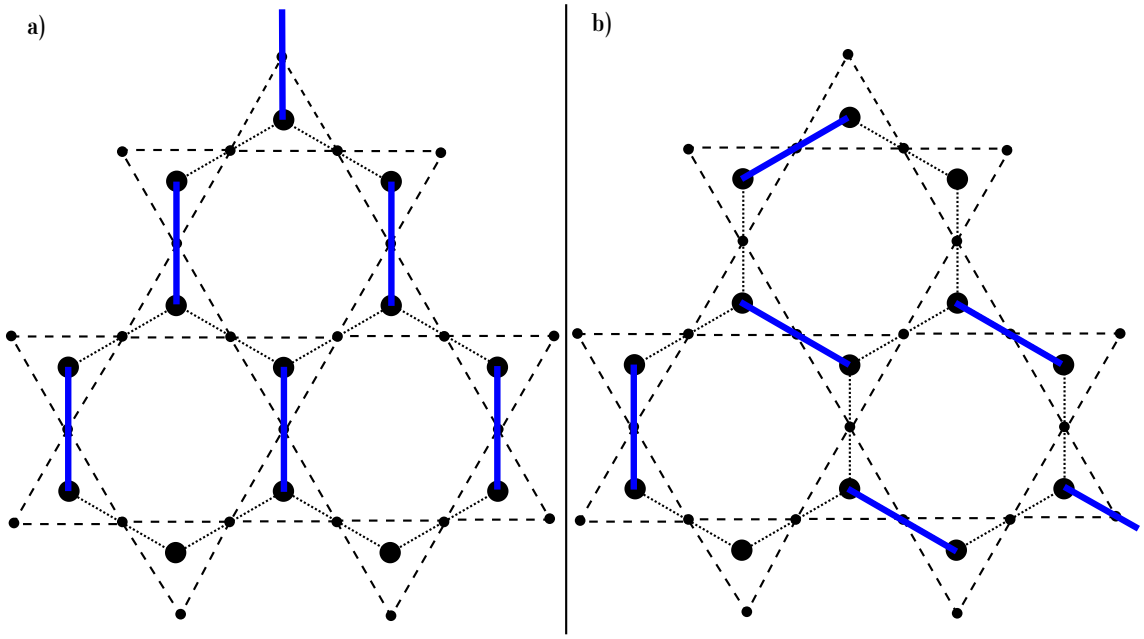

Figure 26: The kagome lattice (dashed lines, small circles) has the honeycomb lattice as its dual (dotted lines, large circles). The local ice rule (two spins in, one out) on each up triangle of the kagome lattice maps exactly onto the placement of a dimer (blue) on the honeycomb lattice if the dimers are placed on top of the out spins. a) shows an ordered arrangement of dimers found below the Kasteleyn transition temperature, b) shows a random arrangement found above the transition; both cases obey the ice rule.

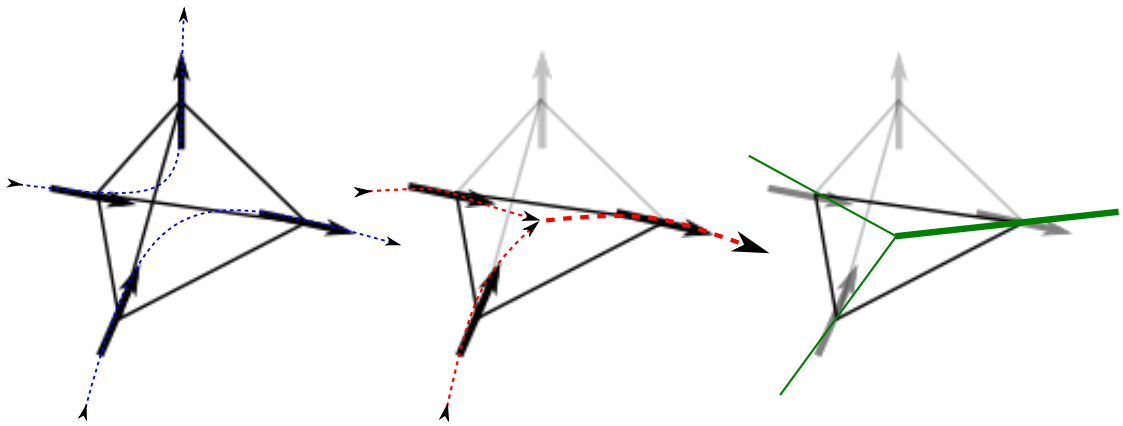

Figure 27: Left. A single tetrahedron of the spin ice lattice obeying the ice rule. As each spin is equally weighted the ice rule produces a divergence free constraint on the spin field (blue). Centre. With a field pinning the central spin (grey) to the $[1, 1, 1]$ direction the ice rule for the remaining three spins is modified to two spins in, one out. An artificial field to which an in spin contributes half the magnitude of an out spin can be defined (red) which retains a divergence free constraint. Right. The topologically constrained kagome lattice is equivalent to hard core dimers on a honeycomb lattice if the out spin is mapped to the dimer position (green).

to that leaving it.

$$\nabla \cdot \mathbf{M}(\mathbf{r}) = 0 \tag{131}$$

The kagome ice model considered in this work is formed by the application of a magnetic field, H, along the $[1,1,1]$ direction of the spin ice model pinning one quarter of the spins perpendicular to the kagome lattice planes, see figure (9), but it is equally valid to apply the magnetic field in the opposite direction parallel to $[\bar{1},\bar{1},\bar{1}]$ as this would simply isolate kagome planes with the pinned spins in the opposite direction. When considering a single kagome plane the divergence free constraint is inherited from the spin ice model but it must be recast onto the three spins that are on each triangle of the lattice. The fourth spin of each tetrahedron is pinned to the $[1,1,1]$ direction and is not able to change its orientation but maintains its role as an 'out' spin so that the new ice rule is two spins pointing into and one out of each up triangle on the kagome lattice, or equivalently two spins pointing out of and one into each down triangle. We define a 'flow' field where the contribution of both inward pointing spins is half that of the outward pointing spin. This artificial flow field, $\mathbf{A}(\mathbf{r})$, once again obeys a divergence free condition as a consequence of the local ice rule as shown in figure (27).

$$\nabla \cdot \mathbf{A}(\mathbf{r}) = 0 \tag{132}$$

If the pinning field was parallel to the $[\bar{1},\bar{1},\bar{1}]$ direction then the opposite direction of the fourth spin on each tetrahedron would enforce an inverse ice rule of two spins out of and one into an up tetrahedron. Thus the choice of the pinning magnetic field determines the ice rule and breaks the Z_2 symmetry of the underlying lattice by selecting one of the two opposite topological sectors. We refer to these as the Z_2^+ and Z_2^- topological sectors and choose the Z_2^+ topological sector at all times in this work.

It is instructive to compare kagome ice to the related system introduced in chapter (1) known as Wills ice in which the ice rule is slightly different [25]. In this model there is no fourth spin associated with each triangle of the kagome lattice and the

ice rule dictates that two spins must be pointing into and one out of *or* two spins must be pointing out of and one into each triangle of the lattice. In this way the Z_2 symmetry of the lattice is maintained as both the Z_2^+ and the Z_2^- topological sectors can coexist on the lattice but there is no longer a topological constraint that can be expressed as a divergence free field condition and hence no Kasteleyn transition occurs. It is possible to define an artificial field in a similar way to the example above but the simultaneous presence of triangles from both topological sectors means that this field would not be divergence free. As Wills ice contains both the Z_2^+ and Z_2^- topological sectors it is symmetric with respect to time reversal; it is the choice of a convention for the direction of time in kagome ice that selects only one of the two sectors.

The correspondence between kagome ice and the dimer model originally studied by Kasteleyn is illustrated in figures (26) and (27). The honeycomb or hexagonal lattice is the dual of the kagome lattice thus each bond of the honeycomb lattice lies between the centres of the triangles in the kagome lattice tracing an identical path to the spin directions of the kagome lattice. The rule governing the placement of dimers on the honeycomb lattice is that a dimer is placed along the bond corresponding to the outward pointing spin on each up triangle which naturally introduces a threefold symmetry to the problem as there are three positions on each up triangle where the dimer may be placed.

When all dimers are placed on the same spin position the lattice is in a topologically long range ordered state and due to the triangular symmetry there are only three such states possible. We refer to these three regions of phase space as continuous topological sectors [101]. Within each continuous topological sector there are a large number of topological states (one of which is the long range ordered state) each of which is composed of one or more microstates. Each state is associated with a quantised topological charge that is directly proportional to the magnetisation of the lattice and can be increased or decreased by one by the topological excitations discussed in the following section. This charge fluctuates about its maximum value, $L = \sqrt{\frac{N}{3}}$ where there are N spins in the lattice, in the region of the continuous topo-

logical sector above the Kasteleyn transition whilst the charges associated with the other two continuous topological sectors fluctuate about 0. It will be shown in the following section that topological excitations carrying no topological charge transform the lattice between microstates of the same topological state whilst topological excitations with a topological charge transform the lattice between topological states of the same continuous topological sector and effect the Kasteleyn transition.

4.3 Topological Excitations

If the topological constraint (the local ice rule) is relaxed then kagome ice becomes a simple paramagnet and the Kasteleyn transition is no longer well defined. In order to remain within the Z_2^+ topological sector in which the Kasteleyn transition exists the topological constraints must be strictly applied by enforcing $J \gg k_B T$ so that there cannot be any single spin flips as any such excitations would violate the local ice rule on a triangle. Simultaneously however the system undergoes a phase transition associated with spin ordering indicating that there must be some method of rearranging spins on the lattice. Rather than a single spin flip, such a lattice update is a continuous loop of spins in the sense that, if the spins are represented vectorially, the head of each spin is adjacent to the tail of its neighbour in the loop. The loops can be categorised into two types, first those that are closed on the lattice, that is, they may evolve across edges of the lattice through periodic boundary conditions but the vector sum of all spins in the loop is always zero; these will be referred to as loops. These loops carry zero topological charge, which is directly proportional to magnetisation, and so alter the lattice configuration without changing the magnetisation. The second type of loop is composed of spins that approximately follow a single direction across the lattice. These excitations close on themselves through the periodic boundary conditions and are referred to as strings; these strings are equivalent to infinite length, infinite energy excitations in the thermodynamic limit and are the lowest energy excitations necessary to update the spin configuration of the kagome lattice whilst remaining in the topological sector defined by the local constraints. Strings are topologically charged excitations as opposed to the topo-

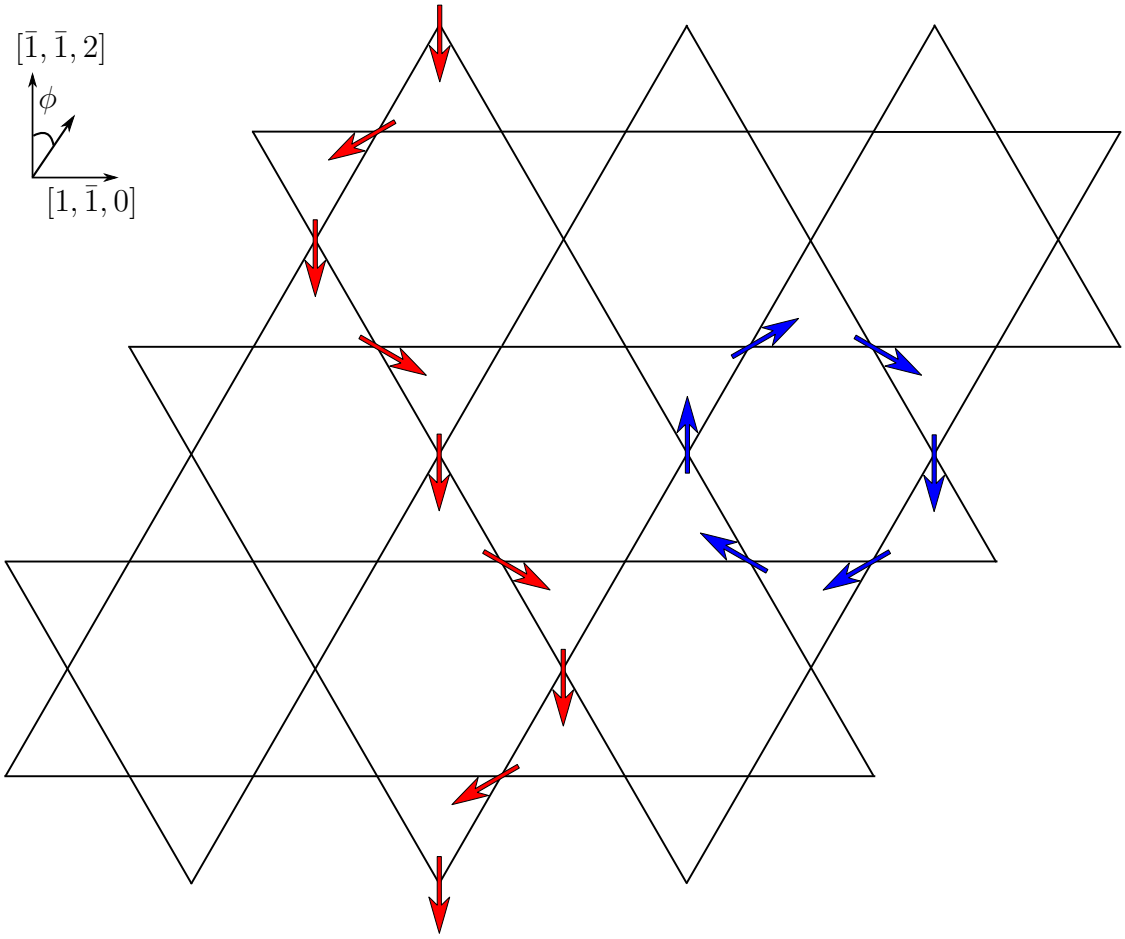

Figure 28: Two possible constructions for a loop through the kagome lattice. The loop marked with red is called an open loop or string as it closes on itself only through the periodic boundary conditions and in the thermodynamic limit it tends toward an infinite length and energy. The loop marked with blue is the smallest possible closed loop on the lattice in the sense that the sum of the magnetic moments around the loop is equal to zero. The $[\bar{1},\bar{1},2]$ direction is marked as is the applied field angle, ϕ, referred to in the text.

logically neutral loops and possess a magnetisation that depends on their length so that when they are placed into the lattice they update both the spin configuration and the total lattice magnetisation. Each string carries a topological charge of one and adding or removing them from the lattice can alter the total topological charge from zero to $L = \sqrt{\frac{N}{3}}$ where L is the maximum number of strings that can be arranged parallel to each other on the lattice. A string and a loop are illustrated in figure (28).

Single spin flips break the topological constraint on a triangle and are not allowed in isolation but they are necessary as components of the topological loop excitations which are constructed using a series of these spin flips. When a single spin flips it

creates a pair of topological defects breaking the ice rule locally and altering the two triangles it is a member of; the spin configuration on one becomes three in (+) and the other becomes three out (−). By flipping a spin on a neighbouring tetrahedron that is 'connected' to the first spin, in the sense that once all spins are flipped it is possible to follow a line through the lattice, it is possible to return the first tetrahedron to a state that obeys the topological constraint and in doing so move one of the defects to the neighbouring tetrahedron breaking the constraint there instead. If the defect is moved across the entire lattice through a series of single spin flips and, through periodic boundary conditions, returns to the other defect then the final spin flip will return both tetrahedra to a state that obeys the topological constraint cancelling the energetic cost of making the initial spin flip but leaving a chain of flipped spins, see figure (29).

The point defects that are created, separated and recombined during the formation of each loop are emergent positive and negative particles within the kagome lattice which take on the character of magnetic monopoles if there are long range interactions present in the system [55]. They are formed by the spatial separation of the magnetic dipole formed during the initial spin flip (see figure (29)) and so can be realised with nearest neighbour interactions only between the spins but they only carry a well defined charge and interact with each other in a coulombic manner in the presence of dipolar interactions [56]. The pinning field for one quarter of the spins on the spin ice lattice restricts the movement of these defects so that they are laterally constrained within each kagome plane until the temperature or field strength is increased to a point at which the kagome ice model breaks down. The movement of each point defect on the lattice is also biased in the presence of any tilt of the field as each consecutive spin flip is associated with a Zeeman energy. This energetic cost determines the favourableness and direction of each loop and will be revisited in the following section where the transition temperature is derived.

Within the kagome ice model it is possible to differentiate between local and non-local dynamics. The local dynamics are due to single spin flip lattice updates which break the ice rules whereas the non-local dynamics are due to the topological

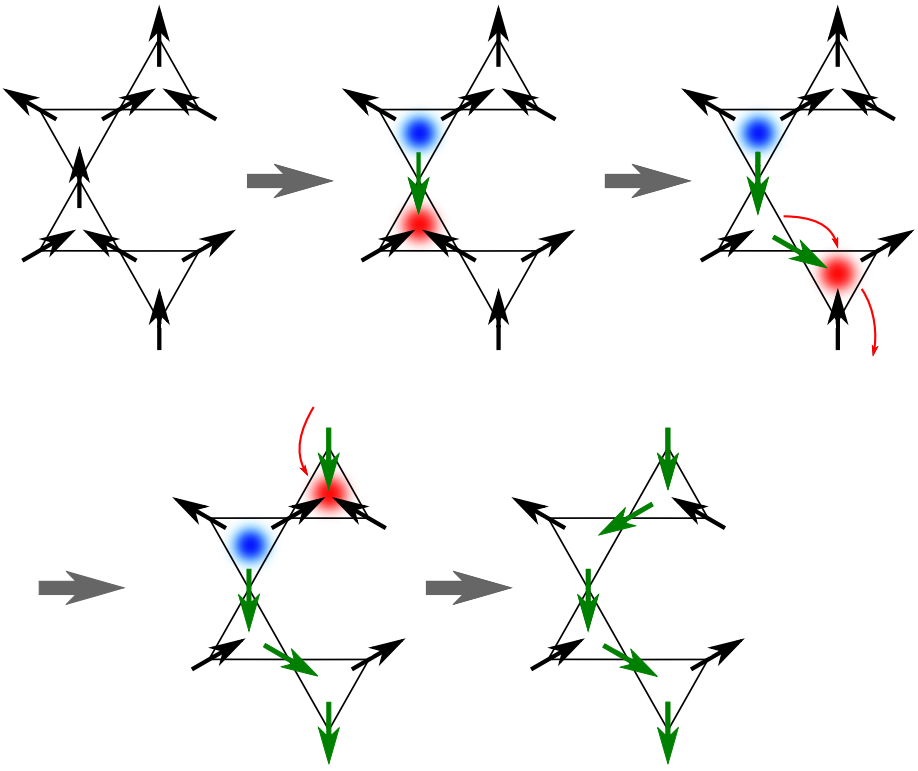

Figure 29: The formation of a topological excitation on the kagome lattice. The upper left lattice configuration is topologically ordered and corresponds to a temperature below the transition temperature. As the temperature is increased above the transition a loop of spins may flip, the process of forming such a loop starts by flipping a single spin (shown in green in the upper central image) to create a positive and negative pair of defects (red and blue). Once the initial defects have been created flipping neighbouring spins spatially separates the defects (red arrows). When selecting the next spin in the loop the string alternates between a unique choice and a choice between two directions as it moves through consecutive up and down triangles. After performing a biased random walk across the lattice and through the periodic boundaries the deconfined defects eventually recombine and anneal each other regaining the energetic cost of the initial spin flip and leaving a trail of flipped spins (green). By considering the loop formation process to occur instantaneously (jumping from the upper left to lower right image) the loop is able to change the lattice order without violating the topological constraints.

excitations which maintain the ice rules. The local dynamics slow down as the temperature is decreased from above the critical point during a simulation and the probability for any spin to flip becomes smaller as the probability depends on a Boltzmann factor for the energy difference associated with flipping that spin. At a characteristic temperature, on the scale of the interaction strength between spins, J, the spins become completely unable to flip and the lattice is frozen into a particular state. At temperatures just above this point only a small percentage of the spins will flip and it is increasingly difficult to sample the system efficiently during a simulation. In contrast to this the creation of topological excitations does not violate the topological constraints hence the internal energy remains constant and there is no barrier to the formation of these excitations. The non-local dynamics do not slow down over the entire temperature range in which the topological excitations are permitted so they can greatly increase the ergodicity of the simulation in the critical region as large numbers of loop excitations persist until temperatures very close to the transition temperature.

The strings provide the lowest energy excitations above the ordered manifold and are able to transform the lattice between topologically ordered and disordered states or into and out of the fully ordered state, whereas the closed loops cannot change the magnetisation of the lattice; they change the spin configuration but their net magnetisation is zero and their role is simply to rearrange the existing lattice configuration. At temperatures just above the transition almost all loops formed during the lattice update procedure are strings which change the magnetisation of the lattice but under conditions increasingly far from the transition closed loops play an increasingly large role and transform the existing strings. They can alter the strings if a portion of the string and the loop overlaps, for example if the string forms one half of a closed loop then when that loop flips the path of the string is changed to the opposite half of the loop, see figure (30). This allows the strings to be mobile on the lattice with the possibility of interacting with each other which is energetically significant as, from the point of view of the entropic contribution to the free energy, two strings passing through the same triangle lose the ability to

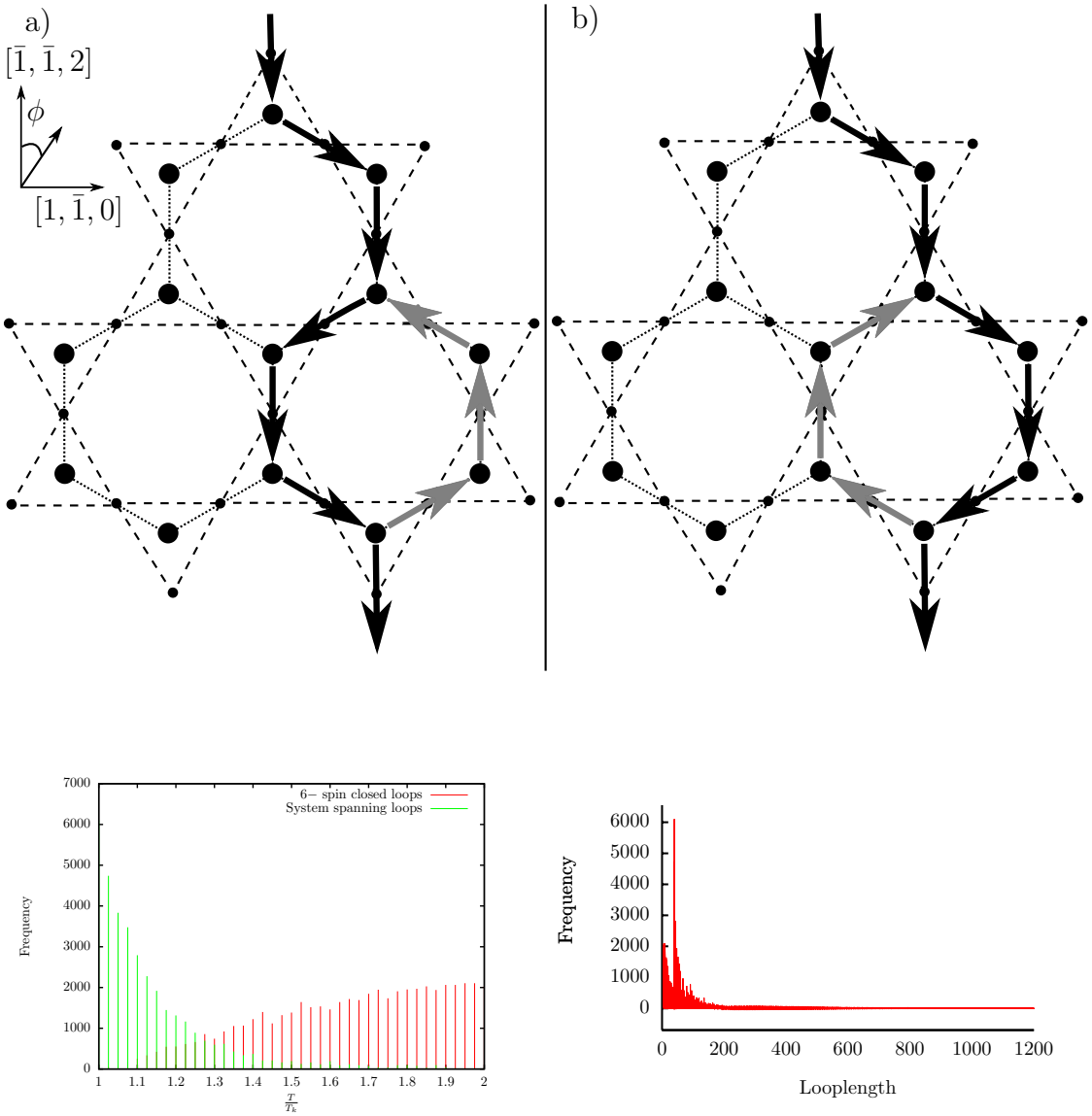

Figure 30: Top: A string traverses the lattice in a linear fashion (black spins). The position of the string can be altered by flipping a closed loop of spins which coincides with a portion of the string. a) and b) show the translation of the string from the left to the right half after flipping the closed loop of 6 spins (grey spins become black and *vice versa*). Bottom left: The frequency of loops containing 6 spins (that alter strings as shown above) and strings that span the lattice as a function of temperature above the transition. Bottom right: Frequency of loops as a function of loop length. The highest frequency of loops is at a loop length corresponding to the lattice width (shown in green on the left) although the left figure shows that these loops are only significant at temperatures close to the transition temperature. Closed loops containing 6 spins (shown in red on the left) become dominant at higher temperatures although the distribution becomes broader with significant numbers of loops of shorter lengths and a small number of loops at all lengths up to the system size present.

choose their direction within that triangle and hence the entropy associated with the choice. As such the string defects repel each other resulting in the continuous transition at the Kasteleyn temperature.

4.4 The Kasteleyn Transition Temperature

In the low energy regime the kagome lattice is completely ordered so that each up triangle on the lattice has the same spin selected as the outward pointing spin which, in the case of continuous topological sector 1, is spin 1. As illustrated in figure (27) the outward pointing spin is equivalent to the dimer position in the dimer representation of the lattice and so all dimers similarly occupy the same position on each triangle. Starting from this long range ordered state corresponding to continuous topological sector 1 and placing a single topologically charged excitation into the lattice alters the energy of the system, and by examining the energetic contributions of this process it is apparent that there is a singular temperature at which it becomes more favourable to include such an excitation than not to include it.

The equivalent to Kasteleyn's chemical potential for the dimers is the Zeeman energy required to take an outward pointing spin and flip it to an inward pointing spin which is the energy required to remove a dimer from its lattice position to an infinite distance away. Defining a field with an angle ϕ relative to the $[\bar{1}, \bar{1}, 2]$ direction as indicated in figure (30) the chemical potentials for the three dimer sites are,

$$\mu_1 = 2\mathbf{H} \cdot \mathbf{S}_\perp \cos(\phi) = \frac{2}{3}\epsilon_y \tag{133}$$

$$\mu_{2/3} = -2\mathbf{H} \cdot \mathbf{S}_\perp \cos\left(\frac{\pi}{3} \mp \phi\right) = -\left(\frac{1}{3}\epsilon_y \pm \epsilon_x\right) \tag{134}$$

where we have defined the quantities $\epsilon_{y/x}$, which are proportional to the resolved components of the field interaction terms, as they will appear again in later deriva-

tions,

$$\epsilon_y = 2\sqrt{2}H\cos(\phi) \tag{135}$$

$$\epsilon_x = \frac{2\sqrt{2}}{3}H\sin(\phi) \tag{136}$$

The fugacities may then be defined as,

$$z_i = \exp(\beta\mu_i) \tag{137}$$

so that a general expression for the partition function is

$$\mathcal{Z} = Tr_{n_1,n_2,n_3} g(n_1,n_2,n_3)\exp(\beta(n_1\mu_1 + n_2\mu_2 + n_3\mu_3)) \tag{138}$$

where n_i is the number of dimers on site i and $g(n_1,n_2,n_3)$ is the degeneracy for fixed n_i. The total number of dimers, n, is fixed at one per up triangle hence we can also define a Helmholtz free energy $\mathcal{F}(T,n,\mu_1,\mu_2,\mu_3)$. For the case under inspection in which the lattice is perfectly ordered with the dimer on site 1 for every triangle we may write a simple partition function for a single triangle,

$$\mathcal{Z}_{tri.} = z_1 + z_2 + z_3 \tag{139}$$

In order to place a string into this ordered lattice the dimer must move from site 1 to either site 2 or 3 on the triangles that the string passes through. As described previously once this has happened on a single triangle it must continue through a series of consecutive triangles in order to join the ends of the string and leave the lattice obeying the topological constraints. Therefore the only requirement at each step in the progress of the string is the probability to move the dimer from position 1 to 2 or 3 on a single triangle only as if this is possible then it is more favourable to flip all other spins necessary to complete the loop than to leave a pair of topological defects on the lattice. The probability to move the dimer to position i on the triangle

is,

$$P_i = \frac{z_i}{\mathcal{Z}_{tri.}} \tag{140}$$

and as the dimer can move stochastically to site 2 or 3 the condition to add the first string into the ordered lattice is,

$$P_2 + P_3 \geq P_1 \tag{141}$$

$$\implies z_2 + z_3 \geq z_1 \tag{142}$$

$$\implies \exp\left(-\beta\left(\frac{\epsilon_y}{3} + \epsilon_x\right)\right) + \exp\left(-\beta\left(\frac{\epsilon_y}{3} - \epsilon_x\right)\right) \geq \exp\left(\beta\left(\frac{2\epsilon_y}{3}\right)\right) \tag{143}$$

$$\implies \exp\left(-\beta\frac{\epsilon_y}{3}\right)\left(\exp\left(-\beta\epsilon_x\right) + \exp\left(\beta\epsilon_x\right)\right) \geq \exp\left(\beta\left(\frac{2\epsilon_y}{3}\right)\right) \tag{144}$$

$$\implies 2\cosh(\beta\epsilon_x) \geq \exp(\beta\epsilon_y) \tag{145}$$

$$\implies \ln(2\cosh(\beta\epsilon_x)) \geq \beta\epsilon_y \tag{146}$$

and the string first becomes favourable at the Kasteleyn transition temperature T_K,

$$\ln\left(2\cosh\left(\frac{\epsilon_x}{k_B T_K}\right)\right) = \frac{\epsilon_y}{k_B T_K} \tag{147}$$

Once a single string has been placed in the ordered lattice there are two possibilities, either it is energetically favourable to place further strings into the lattice and the system will undergo a first order transition, or it requires slightly more energy to place a further string into the lattice and the system undergoes a second order transition. The latter is correct and by considering a simple case with the field angle $\phi = 0°$ it is straightforward to see why this is true. In order to determine the Kasteleyn transition temperature at which a string is able to begin to disorder a fully ordered lattice the Gibbs energy change for the addition of a string to the lattice must be negative. Excluding the energetic cost of the initial spin flip which is recovered in the final move of the construction the internal energy of each triangle is equal within the nearest neighbour model hence the energy for the creation of the string only depends on the Zeeman energy of the flipped spins and the entropy associated with each 'move' in constructing the string. It is most convenient to

consider a 'move' as the flipping of two consecutive spins on the lattice as, from the point of view of the string, it takes two spin flips to move a defect from one down triangle to its neighbour, see figure (29). Each move produces an entropy term due to the choice of two directions the loop may take in each down triangle and a Zeeman energy term due to flipping two spins and the addition of the entire loop to the lattice is additive in the energy of each move. The Gibbs energy for each move is,

$$\delta\mathcal{G} = (\delta E_{Zeeman} - k_B T \delta S) \tag{148}$$

where for a single move,

$$\delta E_{Zeeman} = 3\mathbf{H}\cdot\mathbf{S} \tag{149}$$

$$|\mathbf{S}| = \frac{2\sqrt{2}}{3} \tag{150}$$

$$\delta S = \ln 2 \tag{151}$$

Geometric considerations indicate that unit length spins on the trigonal axes of the pyrochlore lattice have a projection onto the kagome plane of length $\frac{2\sqrt{2}}{3}$ and the allowed spin configurations indicate that when moving the positive defect there is a choice of direction for the forming loop in every down triangle. Figure (29) illustrates that flipping two spins against a field in the $[\bar{1},\bar{1},2]$ direction (vertical on the page) requires a Zeeman energy of $3\mathbf{H}\cdot\mathbf{S}$. For a string to appear, $\delta G < 0$

$$0 > \delta G \tag{152}$$

$$\Rightarrow\ 0 > 2\sqrt{2}H - k_B T \ln 2 \tag{153}$$

$$\Rightarrow\ k_B T > \frac{2\sqrt{2}H}{\ln 2} \tag{154}$$

hence the minimum temperature for strings to appear under these conditions,

$$k_B T_K = \frac{2\sqrt{2}H}{\ln 2} \tag{155}$$

This result can be verified by setting $\phi = 0°$ in equation (147). This approach clearly shows that it only becomes favourable to add a string to the ordered lattice when the Zeeman energy cost is outweighed by the entropy gain of doing so. Recalling that closed loops of spins do not change the topological state of the system but do change the microstate by rearranging the existent strings, it is clear that strings are mobile on the lattice. As this is the case then when a second string is added to a lattice containing only a single string they may interact with each other. It is not possible to combine the paths of two strings through a single triangle so any closed loop attempting to move the second string is restricted as it is not able to position the string anywhere that would overlap with the first and the strings therefore repel each other. Thus the entropy of having two strings in the lattice is slightly less than twice the entropy of a single string and the temperature must increase slightly to compensate for this so that the strings are added gradually to the lattice and the transition is continuous.

In an exact parallel of the dimer ordering problem on the honeycomb lattice, Moessner and Sondhi predicted that tilting the field to increase the statistical weight of one of the three subsets of spins would drive the system continuously toward a Kasteleyn transition which would be clearly observable in the ordering of the spins [33]. The Kasteleyn transition occurs at the point when the statistical weight of any one of the dimer orientations is greater than the sum of the other two which is equivalent to the condition for the dimer fugacities (equation (143)),

$$w_1 \geq w_2 + w_3 \tag{156}$$

and there is a transition from incommensurate to commensurate spin order accompanied by a continuous vanishing of the entropy of the disordered phase. This incommensurate phase is distinct from a high temperature random spin orientation as the lattice remains topologically constrained with the local ice rule satisfied on all triangles of the kagome lattice. The incommensurate phase is a weighted combination of the three long-range ordered dimer configurations which is continuously variable toward complete order in one of the three dimer orientations.

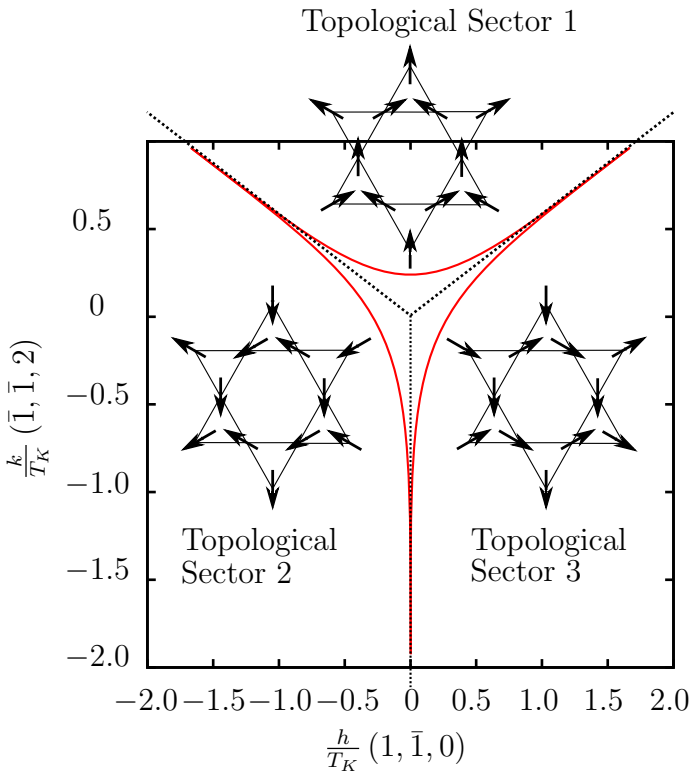

Figure 31: The phase space of kagome ice with respect to an applied magnetic field. As a function of $\frac{H}{T}$ the Kasteleyn transition temperature collapses to the line shown in red within each of the three continuous topological sectors (separated by dotted lines). The long range ordered topological ground state is indicated within each sector. The transition temperature line displayed here is an exact result calculated using equation (147), not a schematic representation.

The equation defining the Kasteleyn transition temperature, (147), does not have a closed form solution (except at $\phi = 0°$) but can be solved iteratively. As the field angle increases the transition temperature decreases until at $\phi = 60°$ this results in $T_K = 0$. The phase diagram of temperature against applied field angle is plotted in figure (32). For applied field angles $|\phi| > 60°$ it is necessary to consider transitions to an equivalent continuous topological sector ordered with a maximum statistical weight $w_{2/3}$ and there is therefore a discontinuous change in the ground state with a degeneracy at the point $\phi = \pm 60°$; the three-fold symmetry of the lattice produces identical behaviour in each sector and so it is only necessary to consider one as illustrated in figure (31).

4.5 Thermodynamics of the Kasteleyn Transition

Within the spin ice nearest neighbour model the internal energy of each tetrahedron on the lattice is equal hence the Helmholtz energy is entirely entropic,

$$\mathcal{F} = -TS(N, M) \tag{157}$$

and, as may be anticipated for an ordering transition, the transition is driven by entropic considerations. A paramagnetic material also has a purely entropic Helmholtz energy, however its behaviour is different as the only way in which to drive the magnetisation toward its maximum value is by reducing the temperature toward zero in which case the entropy also goes to zero with an infinite slope indicating there is no phase transition and forcing the thermodynamic variable to diverge.

$$\frac{H}{k_B T} = -\frac{1}{N k_B T}\frac{\partial \mathcal{F}}{\partial M} = \frac{1}{k_B T}\frac{\partial TS}{\partial M} = \frac{1}{k_B}\frac{\partial S}{\partial M} \to \infty \tag{158}$$

It should be noted that there is only one distinct thermodynamic variable, $\frac{H}{k_B T}$, within the kagome ice model where the magnetic field strength H is proportional to the derivative of the free energy with respect to the magnetisation,

$$H \propto \frac{\partial \mathcal{F}}{\partial M} \tag{159}$$

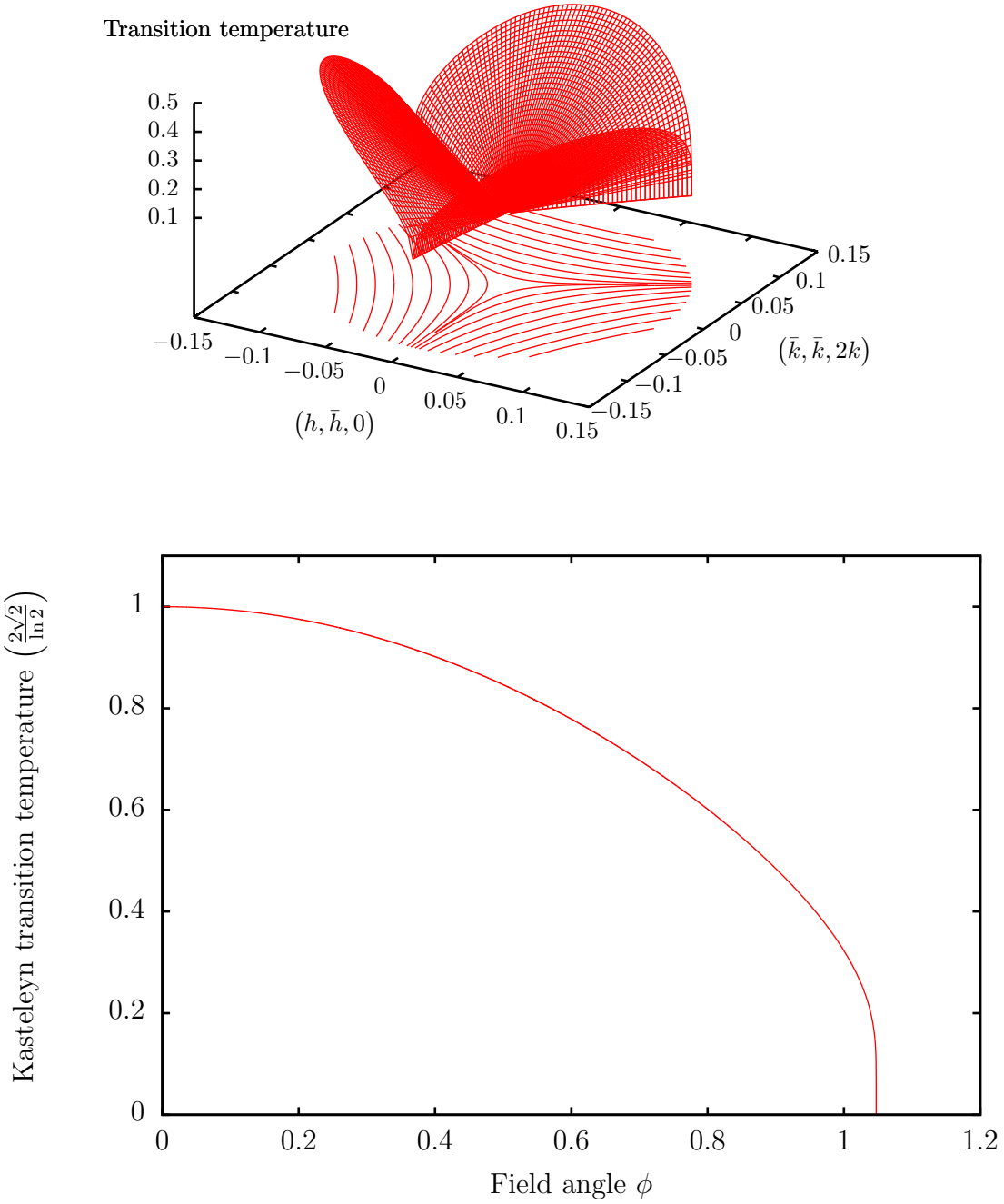

Figure 32: Top: The complete phase space of the Z_2^+ topological sector as a function of the inplane magnetic field. The behaviour of the model is identical in each of the three continuous topological sectors except that the long range ordered state has a different dimer position dominant; at the crossover between these three groundstates the transition temperature falls to zero. For applied field in units of Tesla and and spin moments in units of Joules per Tesla the transition temperature is defined in Kelvin. Bottom: The Kasteleyn transition temperature as a function of the applied field angle in radians for a field of magnitude one within one half of continuous topological sector 1, calculated using equation (147). The transition temperature varies continuously as a function of applied field angle between a maximum at $\phi = 0$ and zero at $\phi = \frac{\pi}{3}$. Varying the field for negative field angle $-\frac{\pi}{3} \leq \phi \leq 0$ produces mirror symmetric behaviour about the line $\phi = 0$.

however for convenience the magnetic field is generally held constant whilst the temperature is varied. The divergence free constraint in kagome ice allows the entropy to go to zero as the magnetisation is driven to its maximum value but this occurs at a finite temperature so that the variable $\frac{H}{k_B T}$ also remains finite and the derivative of the entropy with respect to the magnetisation has a finite slope such that it is possible to define a Kasteleyn phase transition. Kagome ice and other systems with large ground state degeneracies are identified as cooperative paramagnets [102] distinct from simple paramagnets as the internal energy is constant and may be removed from consideration but the system couples to an external field which induces a transition at finite temperature. It is possible to perform a Landau analysis of the transition by defining the Gibbs energy,

$$\mathcal{G} = \mathcal{F} - N\mathbf{M} \cdot \mathbf{H} \tag{160}$$

as the relevant work term for this system is due to the Zeeman interaction. Expanding in powers of the magnetisation difference $m = M_{Max.} - M$ near to the transition,

$$\frac{\mathcal{G}_L}{N} = (H - H_K)m + \frac{\alpha_2}{2}Tm^2 + \frac{\alpha_3}{3}Tm^3 + \cdots - M_{Max.}H \tag{161}$$

where $\mathcal{G}_L$ represents the Landau approximation to the Gibbs energy, H_K is the critical field at a fixed temperature and α_i are a series of constants. In keeping with a typical Landau analysis of a continuous transition, see section (1.3.3), above the transition $\alpha_2 > 0$ is expected to dominate the behaviour of the energy and for fixed field the critical behaviour of the magnetisation is,

$$m = m_0(T - T_K)^\beta, \quad \beta = 1 \tag{162}$$

where T_K is the critical temperature and $m_0 \equiv M_{Max.}$.

Calculating the upper critical dimension using the hyperscaling relation indicates

the regime in which it is valid to use such a mean field approach,

$$2\beta + \gamma = (d-1)\nu_{\perp} + \nu_{\parallel} \tag{163}$$

Evaluating this with mean field values $\beta = 1$, $\gamma = 0$, $\nu_{\perp} = 0.5$, $\nu_{\parallel} = 1$ where the differing perpendicular and parallel critical exponents for the correlation length are a consequence of the model, see for example [33, 103] and the results presented in chapter (6), yields $d = 3$ indicating that in a 2D setting such as kagome ice mean field exponents are not appropriate and will be subject to correction terms [100].

In this system it is possible to calculate the Gibbs energy above the transition analytically. It has a fluctuation driven contribution to the singular free energy that scales to the power of $\frac{3}{2}$, $G \sim (T - T_K)^{\frac{3}{2}}$, which prompted Nagle to classify this as a $\frac{3}{2}$-order transition however this is a correction to the mean field value rather than indicating a critical exponent in its own right and the analysis above indicates that it is a continuous transition. The 2D critical behaviour causes the heat capacity to be divergent from above the transition, $C \sim (T - T_K)^{-\frac{1}{2}}$, with a pseudo critical exponent $\alpha = \frac{1}{2}$ but below the transition the Gibbs energy is constant so that the specific heat jumps discontinuously to zero at the transition, is strictly zero below it and therefore has the critical exponent $\alpha' = 0$ in agreement with mean field theory. The Kasteleyn transition requires careful categorisation as it is asymmetric unlike a typical continuous transition. Below the transition temperature the system is completely ordered and the energy landscape is perfectly flat indicating that there can be no fluctuations, the entropy is strictly zero and the system is a vacuum for fluctuations [104].

The dimer analysis of the transition presented previously provides a general partition function, equation (138), from which it is possible to calculate theoretical values for some common thermodynamic quantities. The particle enthalpy is defined

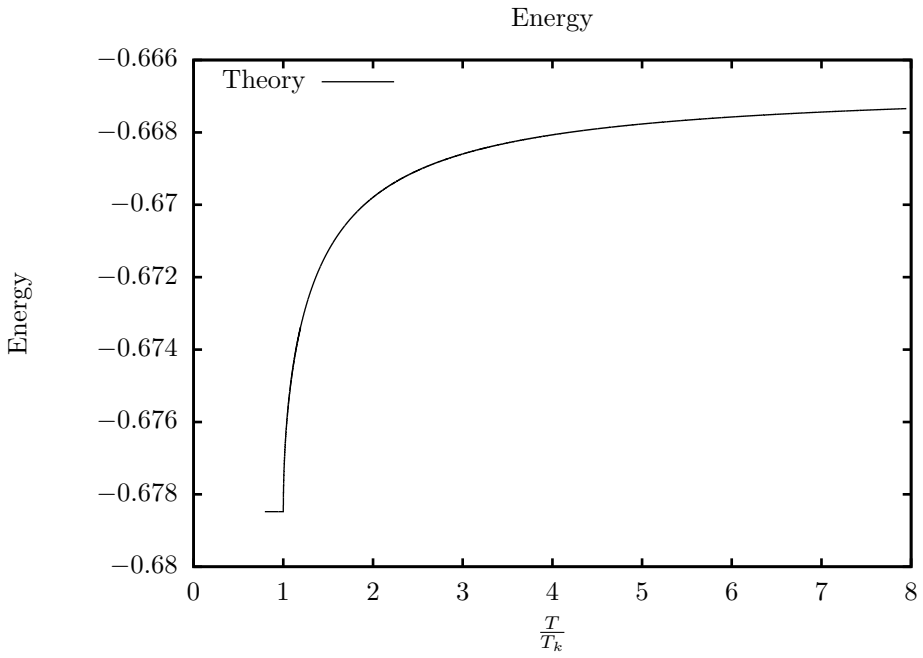

Figure 33: Mean energy per spin for kagome ice with an applied field at an angle $\phi = 20°$.

in the standard way,

$$\mathcal{H} = -\frac{\partial \ln(\mathcal{Z})}{\partial \beta} \tag{164}$$

$$= \mathcal{U} - \sum_i \langle n_i \rangle \mu_i \tag{165}$$

$$= \mathcal{U}_{triangle} - \mu N \tag{166}$$

and the internal energy is constant, $\mathcal{U}_{triangle} = 2$. The mean energy per spin is therefore one third of the sum of the internal energy and the mean energy per dimer,

$$\bar{E} = -\frac{1}{3}(\mathcal{U}_{triangle} + n_1\mu_1 + n_2\mu_2 + n_3\mu_3) \tag{167}$$

as illustrated in figure (33), clearly indicating the asymmetric character of the transition with a strict vacuum for energetic fluctuations below the transition temperature.

Wu provides an explicit expression for the partition function [96],

$$\ln \mathcal{Z} = \frac{n}{8\pi^2} \int_0^{2\pi} d\theta \int_0^{2\pi} d\phi \ln\left(z_1^2 + z_2^2 + z_3^2 + 2z_1 z_2 \cos(\theta) + 2z_1 z_3 \cos(\phi) + 2z_2 z_3 \cos(\theta - \phi)\right) \quad (168)$$

with which it is possible to calculate the mean number of dimers on each site,

$$\langle n_{2/3} \rangle = -\frac{\partial \mathcal{F}}{\partial \mu_{2/3}} = \frac{1}{\beta} \frac{\partial \ln(\mathcal{Z})}{\partial \mu_{2/3}} \quad (169)$$

and as the total number of dimers, n, is fixed $\langle n_1 \rangle$ is not independent,

$$\langle n_1 \rangle = n - \langle n_2 \rangle - \langle n_3 \rangle \quad (170)$$

At high temperatures the dimer positions are randomised and the distribution tends to $\langle n_1 \rangle = \langle n_2 \rangle = \langle n_3 \rangle = \frac{n}{3}$ whereas at temperatures below the transition the lattice has long range order with all dimers on the same position and for the continuous topological sector under consideration $\langle n_1 \rangle = 1, \langle n_2 \rangle = \langle n_3 \rangle = 0$. Defining the fraction of dimers averagely occupying a site as $\alpha_i = \frac{\langle n_i \rangle}{n}$,

$$\alpha_2 = \frac{1}{\pi} \cos^{-1}\left(\frac{z_3^2 - z_2^2 + z_1^2}{2 z_1^2 z_3^2}\right) \quad (171)$$

$$\alpha_3 = \frac{1}{\pi} \cos^{-1}\left(\frac{z_2^2 - z_3^2 + z_1^2}{2 z_1^2 z_2^2}\right) \quad (172)$$

and α_1 is dependant as before,

$$\alpha_1 = 1 - \alpha_2 - \alpha_3 \quad (173)$$

Using these expressions the magnetisation, which is naturally a measure of the dimer

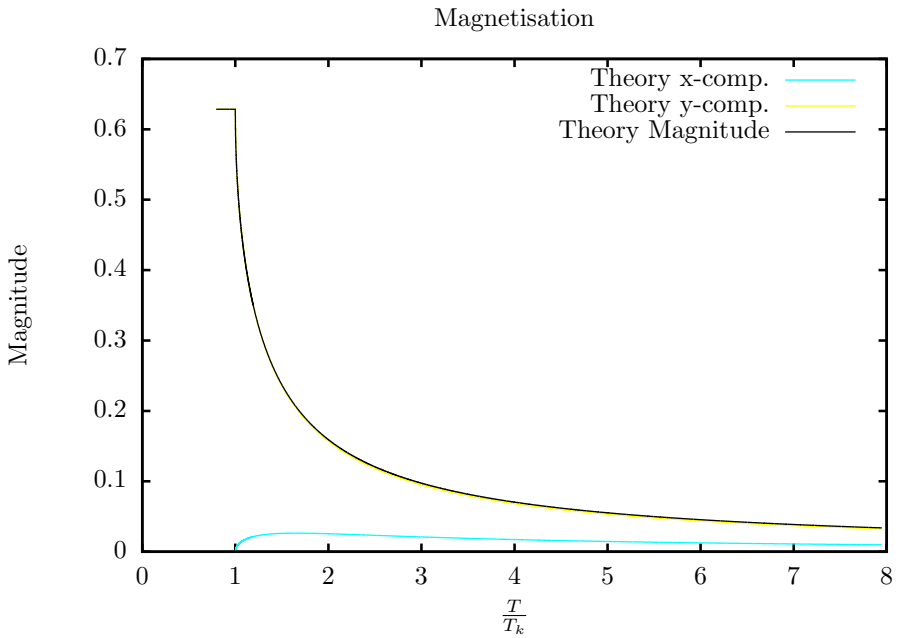

Figure 34: The theoretical magnetisation of kagome ice in a field at an angle $\phi = 20°$. At high temperatures the magnetisation is randomised and tends to a value of zero. As the temperature is reduced the magnetisation increases in both the x and y directions corresponding to the field angle however the x-component disappears at the transition as the long range ordered state that the lattice must obtain at this temperature has a magnetisation parallel to the y-axis only.

occupation on the sites, can be predicted,

$$M_y = \frac{4\sqrt{2}}{9}\left(1 - \frac{3}{2}(\alpha_2 + \alpha_3)\right) \tag{174}$$

$$M_x = \frac{2}{3}\sqrt{\frac{2}{3}}(\alpha_3 - \alpha_2) \tag{175}$$

The magnetisation behaves unusually due to the fact that the long range ordered state is identical for a range of field angles $-60° \leq \phi \leq 60°$. At high temperatures far from the Kasteleyn transition the magnetisation of the lattice aligns with the direction of the applied field but as the temperature is reduced it must eventually order in a direction parallel to the bisector of the continuous topological sector. This biaxial magnetisation is a critical property of kagome ice which will be further analysed in chapter (6). An example of the magnetisation showing the x and y-components (perpendicular and parallel to the $[\bar{1}, \bar{1}, 2]$ direction respectively is shown in figure (34) where the field is at an angle $\phi = 20°$ and the biaxial behaviour is clearly visible in the x-component just above the transition temperature.

5 The Loop Algorithm

In this chapter the loop algorithm that is necessary to simulate the Kasteleyn transition in kagome ice and reproduce the theoretical behaviour derived in the previous chapter will be discussed in detail. This is the core of the kagome ice model I have written to simulate the thermodynamic response and neutron scattering pattern of kagome ice. In order to facilitate a better understanding of the algorithm some of the other features of the kagome ice model which were not discussed during the introduction to the model and are relevant to the simulation will be explained first.

The lowest energy excitations of the topologically ordered lattice found at or above the Kasteleyn transition temperature were shown in the previous chapter to be topological objects that appear as loops of flipped spins. Any simulation of the kagome ice model must therefore employ an algorithm to update the lattice configuration that uses loops of spins rather than single spin flips as in a conventional Metropolis Monte Carlo simulation.

Using a loop update algorithm has other advantages over a single spin flip algorithm in addition to representing an appropriate lattice excitation. The Metropolis single spin flip update algorithm will definitely flip a spin if it lowers the energy of the lattice. In contrast, if it increases the energy, it will only flip if a random number, $0 \leq rand \leq 1$, is less than a Boltzmann factor of the energetic cost of the spin flip, $rand \leq \exp(\beta E_{flip})$. This procedure is useful as it prevents the simulation becoming trapped in local energetic minima during the simulation; however in practice it is limiting in the region of a phase transition, particularly for frustrated systems with highly degenerate groundstates. In this case the lattice will often be close to a particular groundstate so that almost any spin that is selected to flip will only have a small chance of doing so (due to the large Boltzmann factor) and the length of simulation time required to update the lattice to a state uncorrelated with the first grows exponentially.

In the case of kagome ice the groundstate manifold is energetically degenerate and so a simulation should always be able to explore all of the equivalent states, but in order to progress from one to another using the single spin flip algorithm it

is necessary to add and then remove a defect from the lattice making that process comparable to travelling via an excited state. This intermediate state is energetically close to the groundstate and whilst the energy scale of the temperature is significantly larger it does not slow the simulation down. The single spin flip algorithm introduces local dynamics to the model which do slow down due to the finite energy cost of the spin flip in the critical region near to the Kasteleyn transition, but by using an algorithm which only introduces non-local dynamics at no internal energy cost this restriction is removed.

The utility of updates involving more than one spin simultaneously for Monte Carlo simulations was proposed by Swendsen and Wang [105]. Their cluster update algorithm flipped clusters of similar spins rather than connected loops although the concept is applicable in either case as a loop is a form of a spatially extended cluster. Refinements of the cluster and loop approach have produced update algorithms that do not appear to suffer from critical slowing down at all [106, 107]. A loop algorithm therefore greatly increases the ergodicity of a simulation in a region of phase space where the spins are strongly correlated as large numbers of spins are able to alter their configuration at all temperatures above the value set by the correlation scale and so loop algorithms have been used in previous simulations of ice [108] and the spin ice model. The algorithm developed during this work is based on that of references [35, 109]. Before discussing the procedure by which a loop is constructed two important points relevant to the model are now discussed.

5.1 Mean-Field Spin Approximation

The kagome ice model is not strictly 2D as the spins on the pyrochlore lattice have interactions with apical spins above and below the kagome plane and are themselves canted into or out of it. The interactions generated by these elements of the model are vital as they enforce the topological constraints and may not be discarded, see for example figure (27), however for simplicity and computational efficiency it was preferable to avoid creating a 3D model. The lattice is therefore defined as a 2D kagome lattice but each spin is defined as a Heisenberg variable with the z-

component (along the spin ice [111] axis) separated from the x and y components (which lie in the kagome plane). Within the constraint of a 2D lattice there is no dimension remaining in which to include the apical spins however and so it was necessary to represent them with a mean-field approximation. Each spin on the lattice is subject to an effective field term which is equivalent to the presence of an apical spin when averaged over the three spins on any triangle and this consequently subjects the spins in the kagome plane to the full ice rules.

This approximation means there are no discrete objects within the model representing the apical spins but their effects are still present. A consequence of this is that any spin-flipping algorithm is not able to change the direction of this fictitious fourth spin during the MC simulation and so the validity of the model is restricted to the region where an applied field on the [111] axis of the spin ice model strictly pins the apical spin on any tetrahedron to that direction.

5.2 Periodic Boundaries

In order to represent reality as closely as possible it is generally preferable for simulations to include a large number of components. For simulations on small computer clusters the upper limit on this number is determined by the computational time required to simulate the system as this scales with the system size. In the best case this scaling may be linear but often it is worse than this and even the current largest simulations on supercomputers are limited to approximately 10^{11} components although simulation on this scale is also restricted by other factors [110]. Simulations on more accessible computing clusters are necessarily of much smaller systems and so the number of components is significantly different to a macroscopic material containing approximately 10^{23} components and does not represent it well as the proportion of the system experiencing bulk conditions is reduced. This introduces an unrealistic bias to the measured properties of the lattice which is increasingly significant at smaller lattice sizes as the spins on the edges become a greater proportion of the total. The most effective solution to this problem is the use of periodic boundary conditions.

For any simulation in which the structure of the problem is periodic, such as the magnetic lattices encountered in this work, periodic boundary conditions mean that a small portion of the lattice is used to represent its entirety by connecting opposite edges so that any spin on an edge is effectively still surrounded by a full complement of neighbours; this is equivalent to changing a flat planar lattice to a toroidal lattice. For example a spin on the top edge of the lattice has neighbours just below it and the neighbours that would be just above it are on the bottom edge of the lattice. This enables a simulation to be performed in a tractable time using a small lattice whilst still estimating the properties of a much larger lattice.

Connecting the lattice edges with periodic boundary conditions alleviates problems due to lattice edges as all spins have a complete set of neighbours, however periodic boundary conditions are not an ideal solution as they do not create exactly the same conditions as a larger lattice. Properties of the lattice that depend on long-ranged interactions such as spin correlations cannot be accurately replicated because whilst the two-spin correlation continuously decreases with distance in a lattice without periodic boundaries it reaches a minimum and then increases again when they are present because the correlation between identical spin images through periodic boundaries is naturally unity, see figure (35).

Periodic boundaries are particularly relevant in MC simulations of kagome ice as a direct consequence of the topological nature of the defects. A string defect is defined as an infinite length defect in the thermodynamic limit and so any string of flipped spins that is only the length of a simulation lattice without periodic boundaries will be a very poor approximation to this. A string defect must also not break the local ice rule in order to keep the simulation within the groundstate spin configuration. By allowing the strings to connect their ends through periodic boundaries both of these conditions are satisfied.

The Kasteleyn transition is defined energetically as the point at which the entropic gain of placing a string into the topologically ordered lattice outweighs the cost of the Zeeman energy due to flipping those spins, see equation (147). The spins that make up a string will have a variable Zeeman energy depending on their

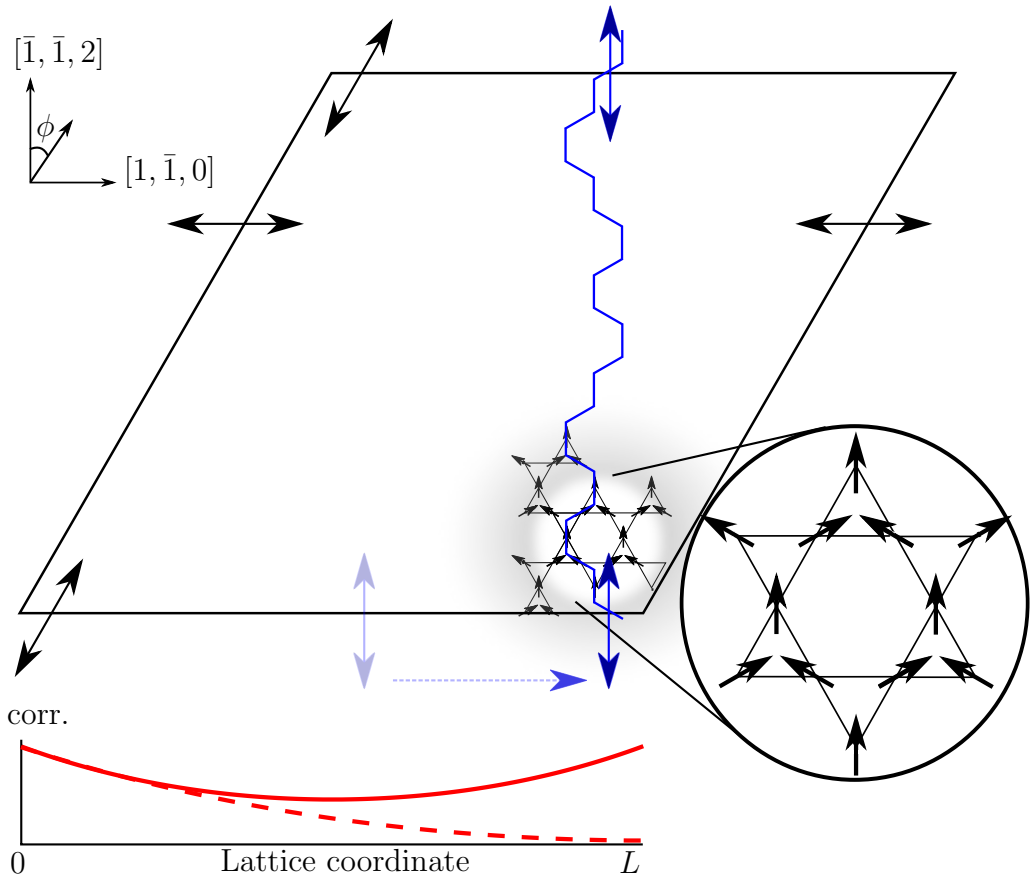

Figure 35: The smallest repeating unit of the kagome lattice is a rhombohedron with internal angles of $\frac{\pi}{3}$ and $\frac{2\pi}{3}$. The black double-headed arrows show how simple periodic boundary conditions connect the edges of a lattice that is this shape. With field parallel to the $[\bar{1},\bar{1},2]$ direction the formation of a string of flipped spins will be approximately parallel to this direction (blue line). In order to reconnect the string to itself through the periodic boundaries so that it has the shortest length the vertical periodic boundaries (blue double-headed arrows) must be 'twisted' by half the lattice width (blue dotted arrow). If the twist was not present the string would reappear at the bottom of the lattice in the position marked by the fainter, leftmost blue arrow and travel across the lattice again before reappearing for the second time near to its origin creating string defects containing approximately twice as many spins. Below the lattice a schematic of the two-spin correlation function is plotted for open (dashed line) and periodic (full line) boundaries showing the correlation does not behave in the same way in the presence of periodic mirror images of the spins.

orientation with respect to the field and the system will preferably place a string with the minimum energetic cost into the lattice. Consequently as the field angle is increased, $0 \leq \phi \leq 60°$, the strings will change their direction to produce the minimum projection onto the field. However, the strings are also constrained by the existing lattice configuration as they can only increase in length by following paths across the lattice that do not create additional topological defects to the initial pair. In the simple case of placing the first string into a topologically ordered lattice with the field parallel to the $[\bar{1}, \bar{1}, 2]$ direction the string is only able to increase its length in a parallel direction on average but may drift a little in the perpendicular direction. At this point it is useful to consider a forming loop as composed of a ballistic trajectory that is determined by the direction of the applied field and a stochastic element that is approximately perpendicular to the ballistic component. In order to accommodate strings of this type that provide the first or last steps out of or into the ordered state the periodic boundary conditions are twisted by half the lattice width so that a string following the $[\bar{1}, \bar{1}, 2]$ direction leaving the top edge of the lattice will reappear on the bottom edge directly below itself and be able to reconnect with itself in the shortest distance as illustrated in figure (35).

5.3 Mapping to a Five-Vertex Model

The kagome lattice can be viewed as a triangular Bravais lattice decorated with triangles of spins at each lattice point. In this case it is necessary to distinguish between the up and down triangles as we have done previously and by selecting for example the up triangles to be centred on the triangular lattice points the down triangles are automatically defined in the spaces between them. In order to define a unit cell which contains one up triangle and regularly tiles the lattice it is necessary to group one down triangle with each up triangle. During the construction of the loop it will be convenient to consider an up and down triangle together so that a single move within the loop algorithm can be defined as a jump of the defect from one down triangle to the next and two spins on the up triangle will be flipped to achieve this. Combined with observation that the central spin on any pair of

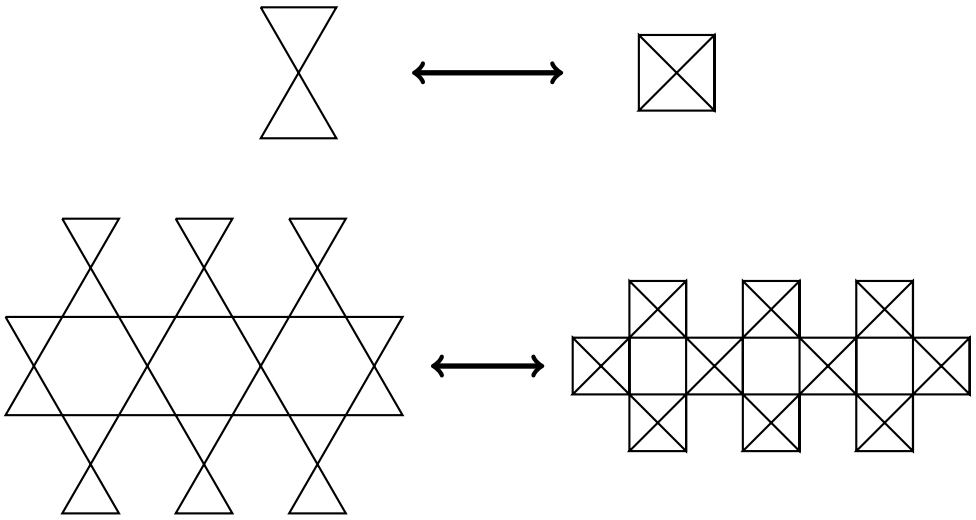

Figure 36: Above: The smallest unit of the kagome lattice that can be repeated to tile the entire lattice is a combination of one triangle with an apical spin above and one with the apical spin below the plane of the lattice. This corresponds to one square of the chequerboard lattice via the five vertex mapping. Below: The central spin on a pair of triangles on the kagome lattice is slaved to its four neighbours and as its orientation is always uniquely defined by the arrangement of the neighbouring spins it does not need to be considered separately. This permits a mapping from the kagome to the chequerboard lattice.

triangles is slaved to the remaining four this suggests a mapping from the kagome lattice to the chequerboard lattice as the orientation of the central spin is always uniquely defined by its neighbours and need not be considered separately [111]. This mapping is displayed in figure (36).

Considered as either a kagome or chequerboard lattice there are only five ways to arrange the spins on a unit of these lattices that satisfy the ice rules hence this is a five-vertex model. The entire model is threefold degenerate as there are three possible pairs of an up and a down triangle on the kagome lattice that could be considered as the minimum repeating unit and all are equally acceptable; they simply define the three topologically distinct sectors of the phase diagram, see figure (31). The spin arrangements shown in figure (37) are those used throughout this work and define the slave spin as parallel to the $[\bar{1}, \bar{1}, 2]$ direction. There is always one vertex in which the slave spin is antiparallel to its orientation in the other four vertices and this is the groundstate spin configuration; the other four vertices can then be considered as the consequence of the four ways a loop can rearrange the spin configuration as it passes through a vertex. The three sectors also correspond

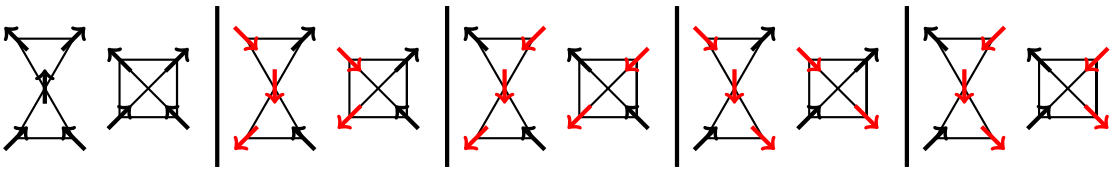

Figure 37: There are five possible spin configurations on a pair of triangles on the kagome lattice which can be defined once the central spin has been chosen (vertical in this figure). Vertex 1 is the groundstate configuration for the topological sector chosen in this work and the red spins on the four other vertices indicate which spins are flipped as a consequence of the four different ways a string can pass through vertex 1. Each vertex is illustrated in both its kagome lattice (left) and chequerboard lattice (right) forms.

to the three possible spins that may be defined as the slave spin but when making the mapping between the kagome and chequerboard lattice this three-fold symmetry is converted to a four-fold symmetry. Although the mapping is exact this requires careful manipulation as quantities that are defined with respect to symmetry, such as an x and y component of magnetisation also require translation to take account of the underlying symmetry change.

5.4 Vertex Probabilities

In the long range ordered topological groundstate considered in this work the lattice is entirely composed of vertex 1, see figure (37) for definitions of vertices. Consider the beginning of the algorithm as the single spin flip of the slave spin creating a pair of defects in the up and down triangle of a single unit cell within this ordered lattice. This definition is convenient but the loop can begin anywhere. The loop then progresses across the lattice via a series of decisions as to whether the loop will move to the left or right leaving the up triangle via spin 2 or spin 3 according to the probabilities P_R or P_L respectively. In between each of these decisions the loop moves through a down triangle in which there is no choice of direction so that there is one stochastic decision per unit cell. The process of determining vertex probabilities for a loop algorithm has been carried out in the complete spin ice lattice [112] and whilst the kagome ice framework is different the following derivation follows similar principles to that work.

Chapter (4) described how the stochastic decision provides the entropic contribution to the forming loop therefore we now evaluate the Zeeman energy contribution as the competition between these two terms governs the transition. The Zeeman energy is evaluated per move and as stated above that involves two spins flipping hence for the left and right moves,

$$\epsilon_R = 2HS_\perp \cos\phi + 2HS_\perp \cos\left(\frac{\pi}{3} - \phi\right) \tag{176}$$

$$\implies \epsilon_R = 3HS_\perp \cos\phi + \sqrt{3}HS_\perp \sin\phi \tag{177}$$

and similarly

$$\epsilon_L = 3HS_\perp \cos\phi - \sqrt{3}HS_\perp \sin\phi \tag{178}$$

where H is the magnitude of the field and the inplane spin component $S_\perp = \frac{2\sqrt{2}}{3}$ so that,

$$\epsilon_{R/L} = \epsilon_y \pm \epsilon_x \tag{179}$$

$$\epsilon_y = 2\sqrt{2}H \cos\phi \tag{180}$$

$$\epsilon_x = \frac{2\sqrt{2}}{3} H \sin\phi \tag{181}$$

just as in equation (147).

In order to express the probabilities in terms of these energies it is necessary to impose two standard conditions upon them. First, the sum of the probabilities for all possible moves at any time must equal unity.

$$\sum_i P_i = 1 \tag{182}$$

$$\implies P_R + P_L = 1 \tag{183}$$

Secondly, due to the sequential nature of the loop construction the algorithm does not satisfy detailed balance, however the probabilities are constructed to obey this

condition, see equation (98) and the explanation above,

$$\frac{P_R}{P_L} = \frac{w_R}{w_L} \tag{184}$$

$$\implies \frac{P_R}{P_L} = \exp(-\beta(\epsilon_R - \epsilon_L)) = \exp(-2\beta\epsilon_x) \tag{185}$$

Using the above conditions the probabilities can be calculated,

$$P_R = \frac{\exp\left(-\beta\epsilon_x\right)}{\exp\left(\beta\epsilon_x\right) + \exp\left(-\beta\epsilon_x\right)} \tag{186}$$

$$P_L = \frac{\exp\left(\beta\epsilon_x\right)}{\exp\left(\beta\epsilon_x\right) + \exp\left(-\beta\epsilon_x\right)} \tag{187}$$

They can also be written in the form

$$P_{R/L} = \frac{1}{2}(1 \mp q) \tag{188}$$

where

$$q = \tanh(\beta\epsilon_x) \tag{189}$$

Combining these descriptions it is possible to write expressions for the Zeeman energy cost and entropic gain per vertex of placing a string into the ordered lattice.

$$\delta E_{\mathrm{Z}} = P_R\epsilon_R + P_L\epsilon_L = \epsilon_y - q\epsilon_x \tag{190}$$

$$\frac{\delta S}{k_B} = -P_R \ln P_R - P_L \ln P_L \tag{191}$$

$$= -\left(\frac{1-q}{2}\right)\ln\left(\frac{1-q}{2}\right) - \left(\frac{1+q}{2}\right)\ln\left(\frac{1+q}{2}\right) \tag{192}$$

Recalling that a loop is only favourable if the free energy associated with it is negative, the transition temperature, T_K is defined using,

$$\delta\mathcal{G} = \delta E_{\mathrm{Z}} - T\delta S \tag{193}$$

$$\implies \delta E_{\mathrm{Z}} = T_K\delta S \tag{194}$$

Using the following identities and the preceding definitions,

$$(1+q)(1-q) = \frac{1}{\cosh^2(\beta\epsilon_x)} \tag{195}$$

$$\frac{1+q}{1-q} = \exp\left(2\beta\epsilon_x\right) \tag{196}$$

$$\beta q\epsilon_x = \frac{q}{2}\ln(\exp\left(2\beta\epsilon_x\right)) \tag{197}$$

the transition temperature is,

$$\frac{\epsilon_y}{k_B T_K} = \ln\left(2\cosh\left(\frac{\epsilon_x}{k_B T_K}\right)\right) \tag{198}$$

exactly equivalent to that calculated in equation (147) providing some reassurance that these probabilities will produce a transition at the correct temperature.

At temperatures above the transition temperature when there are already one or more loops present in the lattice there must also be probabilities for the other situations a newly forming loop may encounter; for example in addition to entering a vertex 1 and requiring a decision to update the position of the head of the loop to the left or the right, the loop may come into contact with a vertex 2, 3, 4 or 5 in which case it has differing options depending on the direction from which it entered the vertex. Choosing to move the positive defect means that the direction of progress of the loop is always 'against' the spin orientation so that in figure (37) the loop will enter a vertex through a spin that is shown pointing out of that vertex. In the case of vertices $2-5$ this means either the black spin in the down triangle or the red spin in the up triangle. If it enters via the black spin the loop cannot continue through the slave spin as it is in the opposite orientation and it must leave again through the other spin on the down triangle which has the effect of exchanging vertex 2 and 3 or vertex 4 and 5 with a probability of one. If it enters via the red spin the loop may either proceed via the slave spin in which case it will return the vertex to the ordered state, vertex 1, or exit via the other spin on the up triangle in which case it exchanges a vertex 2 and 4 or a vertex 3 and 5. The probability to return to an ordered state, P_O, may be calculated by again invoking the detailed

balance condition, equation (98),

$$\frac{P_O}{P_R} = \frac{W_O}{W_R} \tag{199}$$

$$\implies P_O = P_R \frac{W_O}{W_R} = P_L \frac{W_O}{W_L} \tag{200}$$

and using the condition of complete probabilities, equation (182), the probability to exchange vertices when entering an up triangle, P_{ex} is,

$$P_{ex} = 1 - P_O \tag{201}$$

using these definitions $P_O \to 1$ and $P_{ex} \to 0$ as $T \to T_K$ enforcing the transition to the ordered state.

5.5 Construction of a Loop

After deriving the probabilities that determine the path of the loop at the different vertices the construction of a loop can now be discussed in detail. As with the probabilities it is useful to consider the first loop that may be placed into the topologically ordered lattice at a temperature $T = T_K$ as illustrated in figure (38). The procedure is identical at all temperatures but it is more convenient to follow the progress of the loop in an otherwise ordered lattice.

Construction of a loop is initialised by randomly selecting an up triangle and identifying which of the three spins is pointing out of the triangle; in the case of a fully ordered lattice within the current topological sector it will always be spin 1 (labelling is indicated in figure (38)). This spin is flipped which breaks the local topological constraint and creates a pair of topological defects that have been identified as emergent particles with the character of magnetic monopoles [55] as discussed further in section (4.3). Following the simple convention of nearest neighbour only interactions the particles are deconfined and may be spatially separated at no additional energetic cost. This is achieved by flipping one of the other spins on the same up triangle as the first which has the effect of returning the first triangle to its topological groundstate whilst breaking the topological constraint on its neighbour

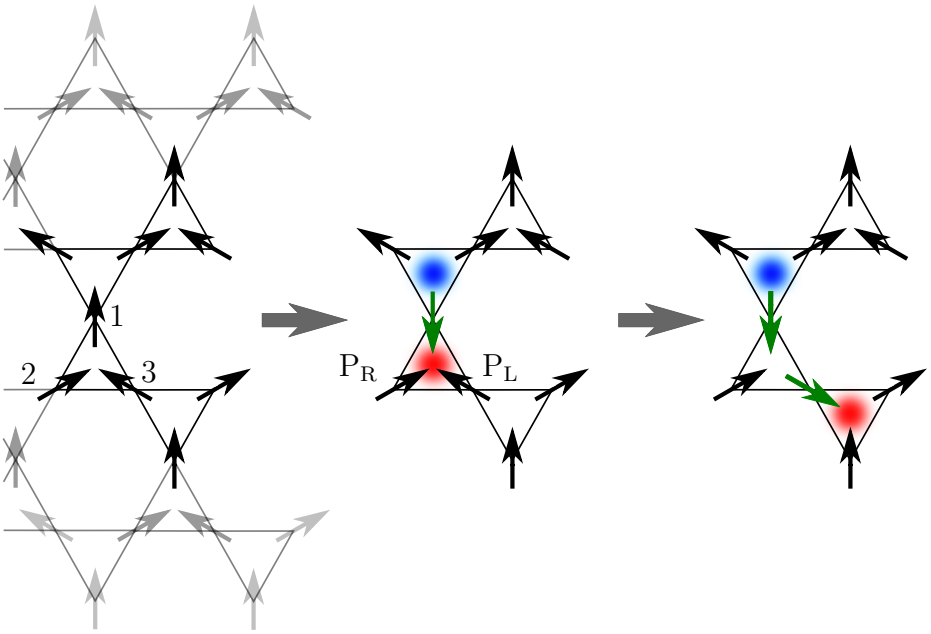

Figure 38: Initialisation of a loop in a fully ordered lattice. In the left image the topologically ordered lattice is shown with the labelling system used to identify the spins on an up triangle. By flipping spin 1 a pair of topological defects are created (central image) which can then be spatially separated by flipping spin 3 (right image). The defect could also have been moved by flipping spin 2, the spins are selected to flip with a probability $P_{L(eft)}$ and $P_{R(ight)}$ with respect to the spin that is currently the head of the loop, in this case spin 1.

so that the defect hops from one triangle to the next. In this model the convention has been chosen so that the positive defect is moved across the lattice whilst the negative defect remains where it is created, however all arguments are equally valid if the negative defect is moved whilst the positive defect remains stationary. In order to prevent the energetic barrier of the initial single spin flip slowing down the algorithm the entire construction of the loop is performed as a virtual move. Once the loop has closed and the energy cost of the initial spin flip has been recouped the loop may then be placed in the lattice based only on the energetic considerations of its Zeeman interaction with the field and its entropy as argued in the derivation of the Kasteleyn transition temperature in chapter (4) and above in the derivation of the loop probabilities.

The finished loop must be a series of spins that follow each other in the sense that the head of one must point at the tail of the next. This limits the possible choices for the next spin in the loop as in alternating triangles there is only one

spin that satisfies this condition, for example the next spin to flip in the right image of figure (38) must be the lower spin of the bottom triangle. In the other half of the triangles there are always two spins that can be chosen as the next member of the loop providing the entropic contribution to the energy of the loop and these are selected using the probabilities for the loop to progress left or right, from the perspective of the current head of the loop, P_L and P_R, which are derived in the previous section.

Under the condition that each triangle in the lattice is in a state that obeys the topological constraints and there are no defects present the loop is then able to perform a directed random walk by repeatedly selecting a new spin to add as head of the loop, alternately with probability one, or with probability P_L or P_R. As this is a 2D random walk on a finite lattice the head of the loop will always find its tail again within a finite time and at this point the final spin flip will annihilate the defect that was created during the initial spin flip leaving the lattice obeying the topological constraint on all triangles in the lattice but having flipped a loop of spins.

This algorithm obeys balance but not detailed balance as during the construction of a loop the spins are flipped sequentially so that the probability that any particular spin flip will be followed by the same spin performing a reverse flip is zero as the next spin to flip must always be a neighbour of the first; however on average the number of moves out of and into any lattice state is equal and balance is satisfied. Many Monte Carlo update algorithms are designed to obey the stricter condition of detailed balance but it has been shown that balance alone is a sufficient requirement [84]. The loop algorithm stipulates that the next spin in the loop must always be a neighbour of the current head of the loop and the loop position must change with every move so that it is impossible to have a spin flip followed by the same spin flipping back to its original orientation and the algorithm does not obey detailed balance; however, the derivation of the probabilities for the different possible moves on the lattice does use that condition, equation (98), as although it cannot be realised via spin flips the relationship between the probabilities and their associated

Boltzmann weights remains relevant.

Finally, once the defect at the head of the loop has performed a random walk on the lattice under the influence of the applied magnetic field and along the paths allowed by the spin configuration it will eventually reach a triangle next to the triangle containing the other initial defect. The next spin to flip moves the defects onto the same position in the lattice where they can annihilate each other and return the lattice to its initial condition of strictly obeying the topological constraint. The trail of flipped spins does not violate this constraint but it does update the lattice from one topological state to another.

The complete code for the loop algorithm, as part of the complete kagome ice program, can be found in appendix (B) in the file `loopmc.f90` and the files referenced therein (electronic document only).

6 Kagome Ice Results

One of the major objectives set out at the beginning of this work was to create a computer simulation model of kagome ice in order to produce thermodynamic and neutron scattering data that would be complimentary to experimental work and aid in understanding the Kasteleyn transition, particularly in kagome ice materials. I have written a program in FORTRAN that simulates the kagome ice model detailed in section (1.4.3) by defining a lattice with rhombohedral unit vectors and unit length spins (equations (57) onward) interacting via a nearest neighbour Hamiltonian (equation (55)). The program operates as a typical Monte Carlo simulation except that rather than a single spin flip algorithm I have written an algorithm which flips a continuous loop of spins in a single move in order to avoid violating the topological constraint of the model. This operates by flipping a single spin and then sequentially flipping adjacent spins to that which creates a pair of deconfined topological defects on the lattice, spatially separates them, and eventually recombines them, leaving a trail of flipped spins as described further in Chapter (5).

A property of the loop algorithm is that it will maintain the topological state in which it operates on the lattice so that any simulation must be initiated with the lattice in a configuration that strictly obeys the divergence free condition of the model, equation (54). In practise this was achieved by starting all simulations in a long range ordered state as this is easy to implement and the loop algorithm is able to equilibrate the lattice quickly to reach an appropriate configuration for a given field and temperature. Simulations were then run as a function of changing temperature to produce data for thermodynamic variables such as the magnetisation, energy, susceptibility and specific heat defined in equations (102) to (105).

This program was also designed to run at a single position in the kagome ice phase space and produce a series of data files containing the lattice configuration. This was achieved by equilibrating the lattice from a long range ordered state as described previously and then writing a file containing the coordinates of lattice position and spin orientation for each spin at intervals separated by a large number of lattice updates to ensure the minimum correlation between configurations. These

data files can then be used to produce simulated neutron scattering maps for kagome ice.

The process of generating the spin configuration files is significantly faster than the process required to produce the scattering maps hence I wrote a separate program to perform this task which allowed me to perform simulations in parallel. This program operates by reading the data files to reproduce a copy of the lattice and then measures the Fourier transformed spin correlations as the wavevector is varied over the required positions of the reciprocal space of the lattice, defined in equation (108). Chapter (3) details the methods by which this produces the scattering intensity function, equations (119) and (120), which may then be plotted over a plane of reciprocal space to generate a scattering intensity map or on a single line of reciprocal space to examine the peak shape of the scattering function. The density of points at which to take measurements determines the resolution of the final data and scales as the square of the number of spins in the lattice. This is the dominant factor controlling the simulation time as each point requires the program to loop over all spins in the lattice whilst obtaining the measurement hence a compromise between time and quality was struck and the scattering maps presented here were simulated using a lattice of 10800 spins with 50 different spin configurations which took around 55 hours to process per map.

6.1 Neutron Scattering

The neutron scattering map which is the starting point for all the maps subsequently produced is that with no inplane field, or equivalently with field aligned perfectly along the $[1, 1, 1]$ direction, shown in figure (39). This figure shows the unpolarised, inplane only and pseudospin spin components of the scattering map for kagome ice in a high temperature, zero field, disordered state (at the origin of the phase diagram shown in figure (31)) where the six-fold symmetry and pinch point singularities can clearly be seen. The unpolarised scattering map is formed by considering the total component of each spin which is perpendicular to the scattering vector whilst the inplane map is formed from the component of each unpolarised spin resolved

onto the kagome plane and the pseudospin map is formed from the complementary component, that resolved perpendicular to the kagome plane. The resolution of the spin components perpendicular to the scattering vector and the kagome plane is referred to as a pseudospin component as it can be represented by a variable at each position that simply takes a value of ± 1. Experimentally this component would be observable using polarised neutrons whilst using unpolarised neutrons would produce the scattering map referred to here as unpolarised.

With the field aligned along the $[1, 1, 1]$ direction there can be no Kasteleyn transition regardless of temperature and the system should display its maximum symmetry. In principle, in the absence of a field it is possible to calculate these spin correlations analytically and produce theoretical neutron scattering maps however it is a complicated calculation and prone to error. In this scenario using a simulation model provides a simpler route to the data and once there is a magnetic field present which can drive the system through a Kasteleyn transition the calculation of the correlation functions is significantly harder or perhaps intractable hence it is valuable to employ a simulation based approach from the outset.

Moessner and Sondhi have attempted to calculate this data analytically, see figures (5) and (6) of reference [33]. Their scattering maps should be identical to the pseudospin and unpolarised components of figure (39) and are reproduced to a matching scale below them in figure (40) for comparison, see appendix (A) for detail of the units used in both sets of figures. Figure (40) shows clear similarities but some inversions in scattering intensity between the maps suggesting there was an error present in one or both of the approaches; however favourable comparison with experiment [44] and other simulations [113] gave early indications that our simulations were correct and there was a sign error in their calculation.

6.1.1 Analytic Calculation of the Structure Factor

Without clear knowledge that our program was operating correctly any subsequent results would not stand up to scrutiny thus it was vital to understand the difference between the scattering maps in figure (40) and to do this we examined the correlation

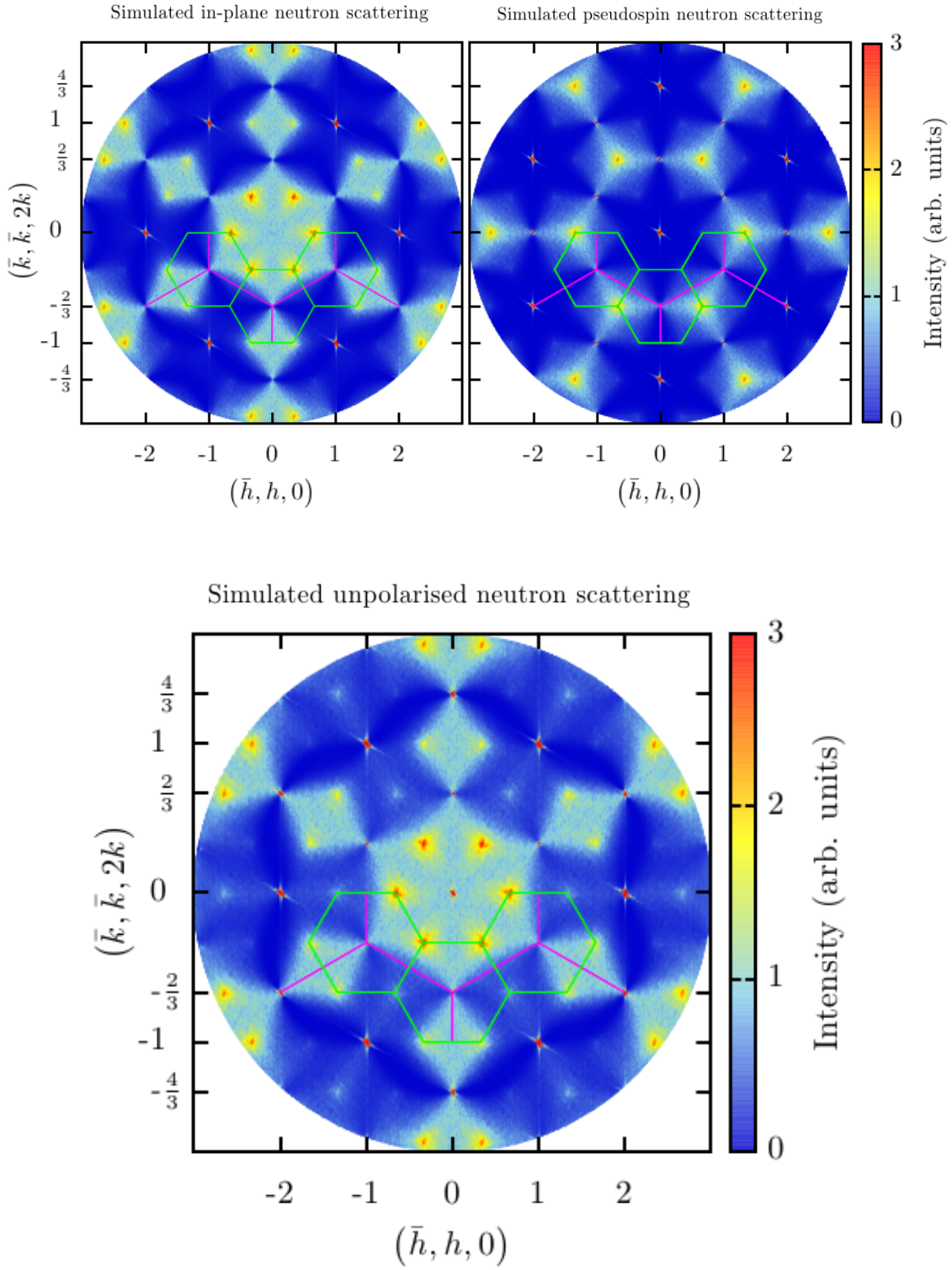

Figure 39: Unpolarised (centre), inplane (top left) and pseudospin (top right) components of the neutron scattering map for kagome ice with no inplane field. In each case the six-fold symmetry of the underlying lattice is clear and there are many positions with obvious pinch point singularities such as $[\frac{2}{3}, \frac{2}{3}, \frac{4}{3}]$ and the 5 symmetry related positions. The unpolarised spin component is composed from a weighted combination of $\frac{8}{9}$ of the inplane and $\frac{1}{9}$ of the pseudospin components according to the manner in which the unit length spin is decomposed into perpendicular and parallel components to the $[1, 1, 1]$ direction. In these and all other scattering maps the Brillouin zone boundaries are shown in the bottom half of the pattern for the kagome (green) and pyrochlore (purple) lattices.

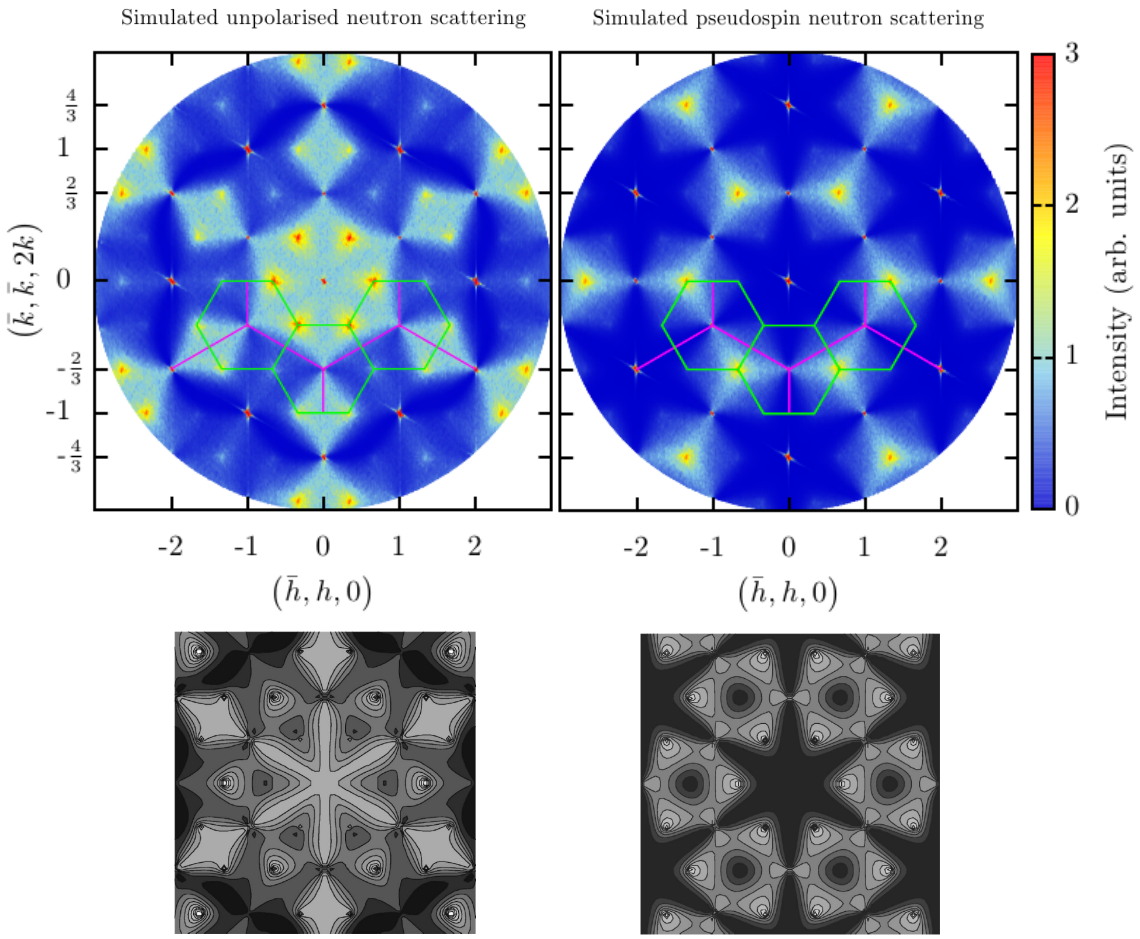

Figure 40: Comparison of the simulated unpolarised (left) and pseudospin (right) neutron scattering maps produced in this work (top) with the analytic calculation presented in reference [33] (bottom). The analytic maps have been scaled to the same size as the simulated data and by examining the maps together it is clear that there are discrepancies between the areas of high and low intensity. This is due to a sign error in the analytic calculation as the simulated maps match both the experimental scattering patterns [44] and other simulated scattering maps such as those in reference [113].

functions used to produce the analytic data.

The equation used to calculate the pseudospin correlations within kagome ice in reference [33] is,

$$c_{\kappa\lambda}(\mathbf{r}) = \langle \sigma_\lambda(\mathbf{r})\sigma_\kappa(0)\rangle - \langle \sigma_\lambda\rangle \langle \sigma_\kappa\rangle \tag{202}$$

where $\mathbf{r}$ labels the location of a triangle within the lattice, λ and κ label the spins within the triangle and the pseudospin variable $\sigma = \pm 1$. This is closely related to the full spin correlation function,

$$C_{\kappa\lambda}(\mathbf{r}) = \langle \mathbf{S}_\lambda(\mathbf{r})\mathbf{S}_\kappa(0)\rangle - \langle \mathbf{S}_\lambda\rangle \langle \mathbf{S}_\kappa\rangle \tag{203}$$

Using these expressions it is possible to calculate exactly the short distance spin correlations and then take the Fourier transform of these to plot the exact structure factor. Although there are nine possible spin to spin correlations between the three spins on each triangle there are only two distinct correlators, that between spins of the same number, and that between spins of different number thus the following expressions encapsulate all the behaviour.

$$c_{11}(\mathbf{r}) \sim \frac{1}{2\pi^2 r^2}\left(\cos\left(\frac{4\pi x}{3}\right) - \cos\left(2\omega\right)\right) \tag{204}$$

$$c_{12}(\mathbf{r}) \sim \frac{1}{2\pi^2 r^2}\left(\cos\left(\frac{4\pi x}{3} + \frac{4\pi}{3}\right) - \cos\left(2\omega + \frac{4\pi}{3}\right)\right) \tag{205}$$

where

$$\tan(\omega) = \cot(\phi) \tag{206}$$

relates their inplane angle, ω to that defined in this work, ϕ. $r = |\mathbf{r}| = \sqrt{x^2 + y^2}$ is the distance between two triangles of the kagome lattice and $\hat{x}$ and $\hat{y}$ lie along the $[\bar{1}, 1, 0]$ and $[\bar{1}, \bar{1}, 2]$ directions respectively. Section (3.2.1) illustrates that the topological constraint within the model is equivalent to the presence of dipolar spin correlations and this is also present in the analytic correlation functions due to their decay as $\frac{1}{r^2}$, which is acceptable in the long distance limit of $\mathbf{r}$ but it is clear that at

$\mathbf{r} = 0$ they are divergent. We resolved this problem by substituting explicit values of the correlators at $\mathbf{r} = 0$ and testing their influence on the scattering map.

By varying these values it was apparent that the single triangle correlations between neighbouring spins were the dominant term in the entire structure factor. In light of this we calculated the correlations within a single triangle only approximating the correlators as $c_{11} = 1$, $c_{12} = -\frac{1}{2}$ to reveal the influence of these shortest distance correlations on the overall spin correlations. Although this is a rather artificial quantity, comparison of the scattering produced with the complete pseudospin map and the equivalent figure of reference [33], see figure (41), shows that single triangle correlations indeed form the basis of the overall result. It is clear that the longer range correlators simply modify the triangular structure factor for example refining the circular scattering at small $\mathbf{Q}$ to a six-pointed star shape. The nearest neighbour spin correlations are enough to enforce the sixfold symmetry of the scattering pattern and define the approximate regions of high and low intensity so that increasingly smaller contributions from longer ranged spin correlations do not transform the single triangle correlations into the analytic calculation of reference [33]. In addition the scattering from the nearest neighbours only agrees with both experimental [44] and other simulated data [113, 114] thus we feel confident that the simulation written for this work is correct whilst the 'form factor' calculated in reference [33] is incorrect and contains some sign errors.

6.1.2 Peak Intensities

In the disordered state at either high temperature or zero field figure (39) shows the system has both Bragg peaks and peaks in the diffuse scattering which are not due to Bragg scattering. Bragg peaks are present even above the transition because the ice rule of two spins in and one out on each up triangle in the lattice means that there is always a net positive spin projection along the $[1, 1, 1]$ direction on these triangles and a negative projection on the down triangles which are necessarily ordered due to the lattice structure, see figure (9) for an illustration of the 3D spin configuration which shows the net projection on the kagome triangles. These ordered

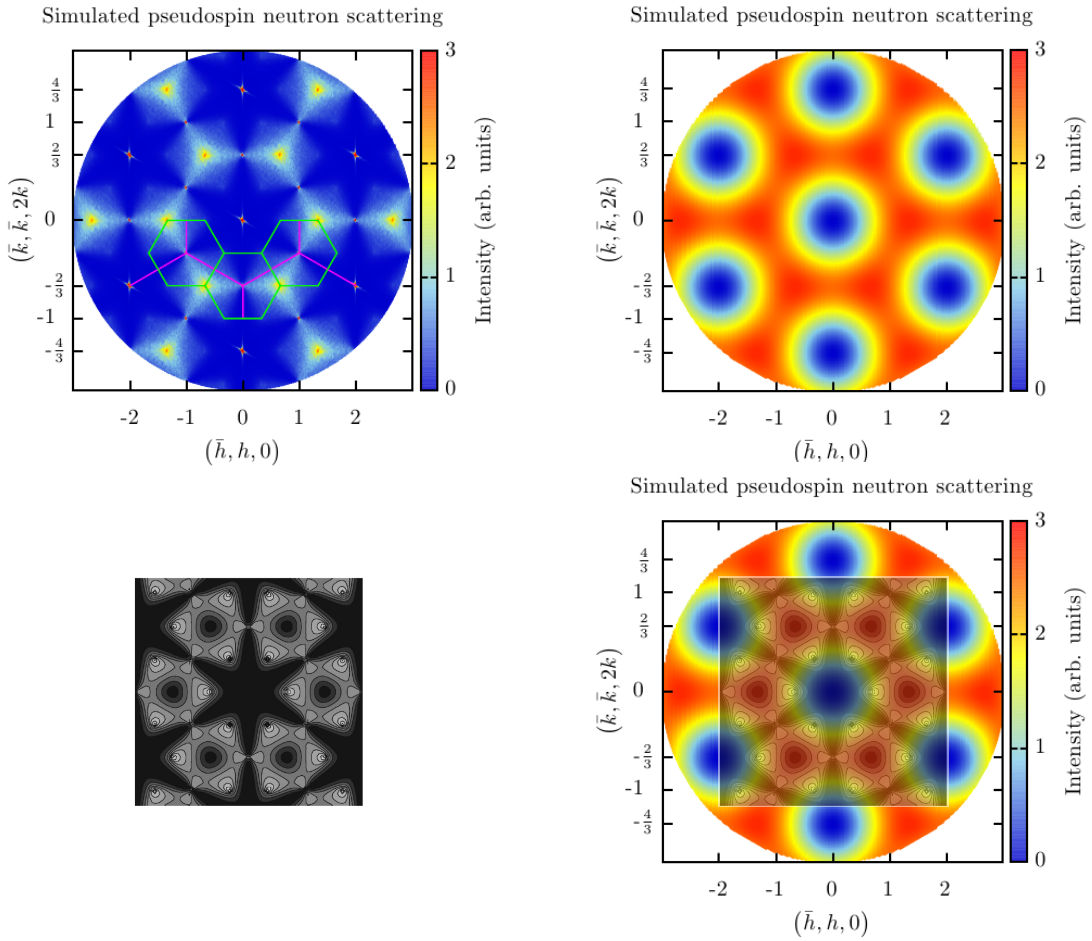

Figure 41: Top left. The full pseudo spin neutron scattering map calculated using the computer simulation developed in this work. Bottom left. The equivalent scattering pattern calculated analytically in reference [33] scaled to the size of the pattern above. The bottom image and the centre of the image above should be identical. Top right. The analytically calculated structure factor for a single triangle only. The maxima and minima are already consistent with the complete pseudospin scattering pattern indicating that the correlations between the three spins on each triangle dominant the complete scattering pattern. Bottom right. Superimposing the analytically calculated full scattering pattern on to the analytically calculated single triangle pattern shows that whilst the large central minima matches for both, the six smaller minima surrounding it on the full scattering pattern are maxima in the single triangle pattern (as they are in the simulated complete pattern). It is clear that adding corrections to account for an increasing number of spin correlations can transform the single triangle pattern into the simulated full pattern but not the analytically calculated full pattern.

spin components produce Bragg scattering at all temperatures even though the spin configuration is disordered within the kagome plane. The scattering intensity is defined (equations (119) and (120)) in the simulation as,

$$S(\mathbf{Q}) = \frac{1}{N}\left\langle \left(\sum_{\mathbf{r}} \mathbf{S}_{\mathbf{r}} \exp\left(i\mathbf{Q}\cdot\mathbf{r}\right)\right)\cdot\left(\sum_{\mathbf{r}} \mathbf{S}_{\mathbf{r}} \exp\left(-i\mathbf{Q}\cdot\mathbf{r}\right)\right)\right\rangle \tag{207}$$

so that the intensity of the Bragg peaks scales linearly with system size which is the expected behaviour for this type of scattering peak however a consequence of the divergence free topological constraint and its implied pseudodipolar interactions is peaks in the diffuse scattering that are not due to long range order and have an intensity which scales logarithmically with system size. Figure (42) shows the maximum intensity of one of each type of peak in the scattering map for varying system sizes and they match the prediction very well. This logarithmic dependence is a mathematical consequence of the dipolar interaction term as

$$\int_1^N \frac{1}{r^2} r\, dr = \ln N \tag{208}$$

The logarithmic peaks are further modified by a structure factor such that their intensity depends on the spin component used to generate the scattering pattern. The pseudo spin scattering patterns are generated by the spin component parallel to the $[1,1,1]$ direction which has a magnitude of $\frac{1}{9}$ and the inplane scattering patterns are generated by the perpendicular components to this which have a magnitude of $\frac{8}{9}$, see equation (71). The full unpolarised pattern is therefore the weighted sum of $\frac{8}{9}$ of the inplane pattern and $\frac{1}{9}$ of the pseudospin pattern.

6.1.3 Scattering Maps for $\phi = 0$

The simplest case in which an inplane field can drive the system through a Kasteleyn transition is when the applied field is along the $[\bar{1},\bar{1},2]$ direction which is parallel to the long range ordered state reached at the transition so that the inplane field angle from the $[\bar{1},\bar{1},2]$ direction $\phi = 0°$. In this scenario we realise for the first time the predictions of Moessner and Sondhi in reference [33] that there are peaks

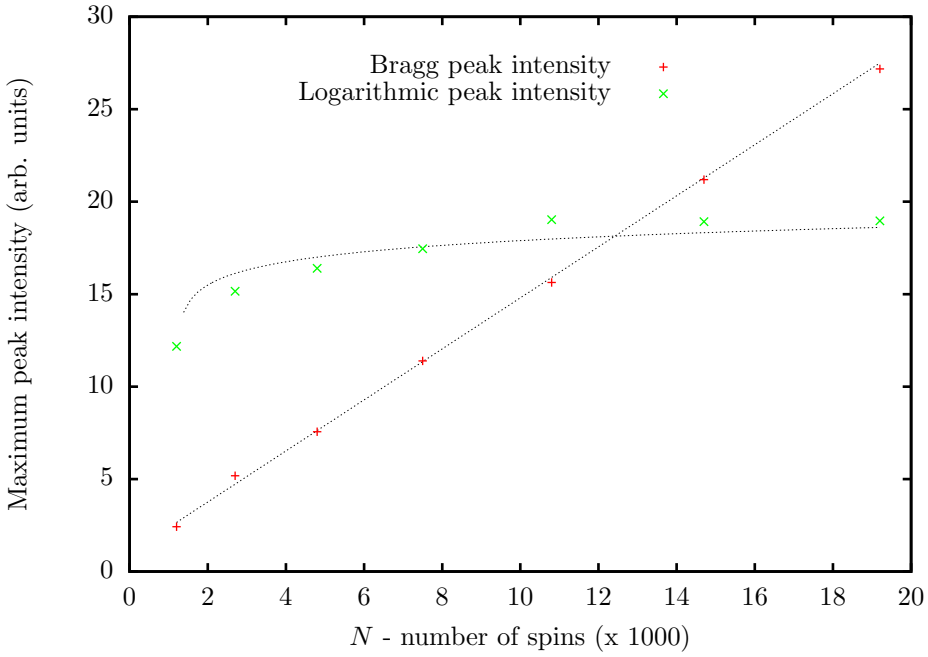

Figure 42: The maximum intensity of the neutron scattering peaks is predicted to either scale linearly with system size or as the logarithm of system size depending on whether the peak is due to Bragg scattering or scattering from pseudodipolar correlations. Analysis of the peak intensities shows that there are clearly two regimes in the peak intensities that match this prediction. With regard to figure (39) the Bragg peak data is taken from the peak at $\mathbf{Q} = (1, \frac{1}{3})$ and the logarithmic peak from $\mathbf{Q} = (\frac{5}{3}, \frac{1}{3})$. Dotted lines are guides to the eye for linear and logarithmic behaviour.

in the scattering which drift continuously with applied field reaching the centre of the Brillouin zone at the transition and that the correlation lengths are parallel and perpendicular to the applied field with critical exponents $\nu_\perp = \frac{1}{2}$ and $\nu_\parallel = 1$. Figure (44) shows the manner in which the peaks drift as a function of the applied field which was previously unverified. Here we have changed the labelling $\nu_x \to \nu_\perp$ and $\nu_y \to \nu_\parallel$ with respect to the applied field direction in preparation for fields that are not parallel to the y direction. This point is returned to in the finite size scaling analysis of the magnetic susceptibility later in this chapter where the values of these critical exponents will be verified.

The application of a field breaks the six-fold rotational symmetry of the pattern reducing it to a two-fold rotational symmetry with mirror symmetry along the x and y directions. The asymmetry of the transition is apparent as when the system reaches the transition all diffuse scattering disappears leaving only the Bragg scattering peaks due to the perfectly long range ordered lattice with triangular symmetry, see figure (43). This is indicative of the asymmetric behaviour where the system is completely frozen below the transition with no fluctuations but is disordered above the transition. This behaviour is most easily visualised as a short 'animation' of the scattering pattern responding to decreasing temperature in the presence of an inplane field. As the thermodynamic variable is the ratio $\frac{H}{T}$ the initial high temperature state is achieved with zero field and then the temperature is reduced whilst the field magnitude is held constant driving the system from a disordered state to long range topological order via the Kasteleyn transition as shown in figures (44), (45) and (46) (electronic version only, animation functionality only available when viewed with Adobe Reader, print version shows representative images of the evolution).

As the temperature is reduced toward the transition the scattering patterns develop two mirror symmetry axes parallel and perpendicular to the field and long range ordered moment direction which reduces the six-fold rotational symmetry to two-fold. The intensity of the background diffuse scattering reduces and appears to be concentrated onto the two high symmetry directions of the lattice at $\phi = \pm 60^\circ$ to

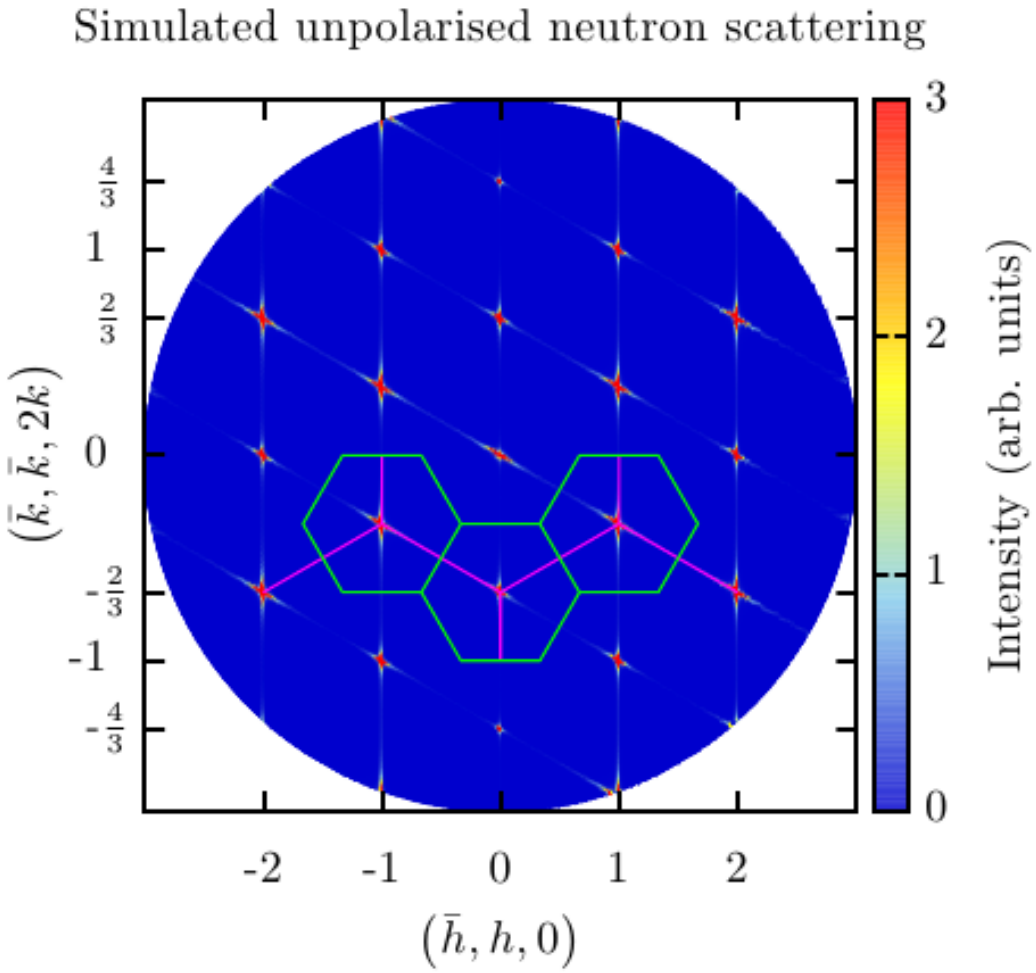

Figure 43: The spin configuration on each triangle of the kagome ice lattice is identical when the system is in the long range ordered state beneath the transition. Each triangle of three spins may then be represented as a single scattering object and the pattern produced contains only the Bragg peaks associated with a triangular lattice however the structure factor still imposes a varying intensity on the different peak positions.

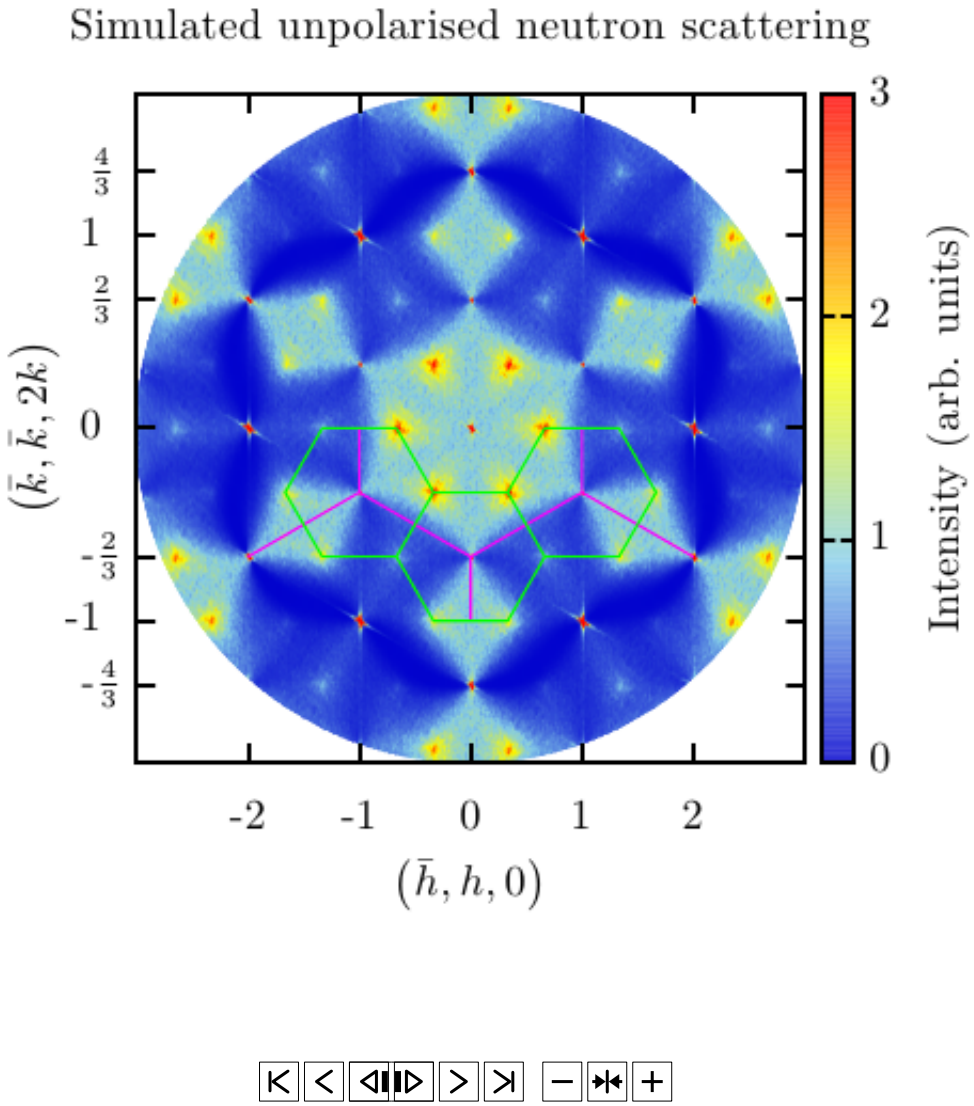

Figure 44: The evolution of the unpolarised component of the neutron scattering pattern of kagome ice in a field at an angle $\phi = 0°$. The thermodynamic variable is increased from $\frac{H}{T} = 0$ to $\frac{H}{T} > \frac{H_K}{T_K}$ to induce a Kasteleyn transition. In the electronic version of this document an animation is displayed whilst in the paper copy representative images from the animation are shown where, apart from at the high temperature starting point, the field was held constant.

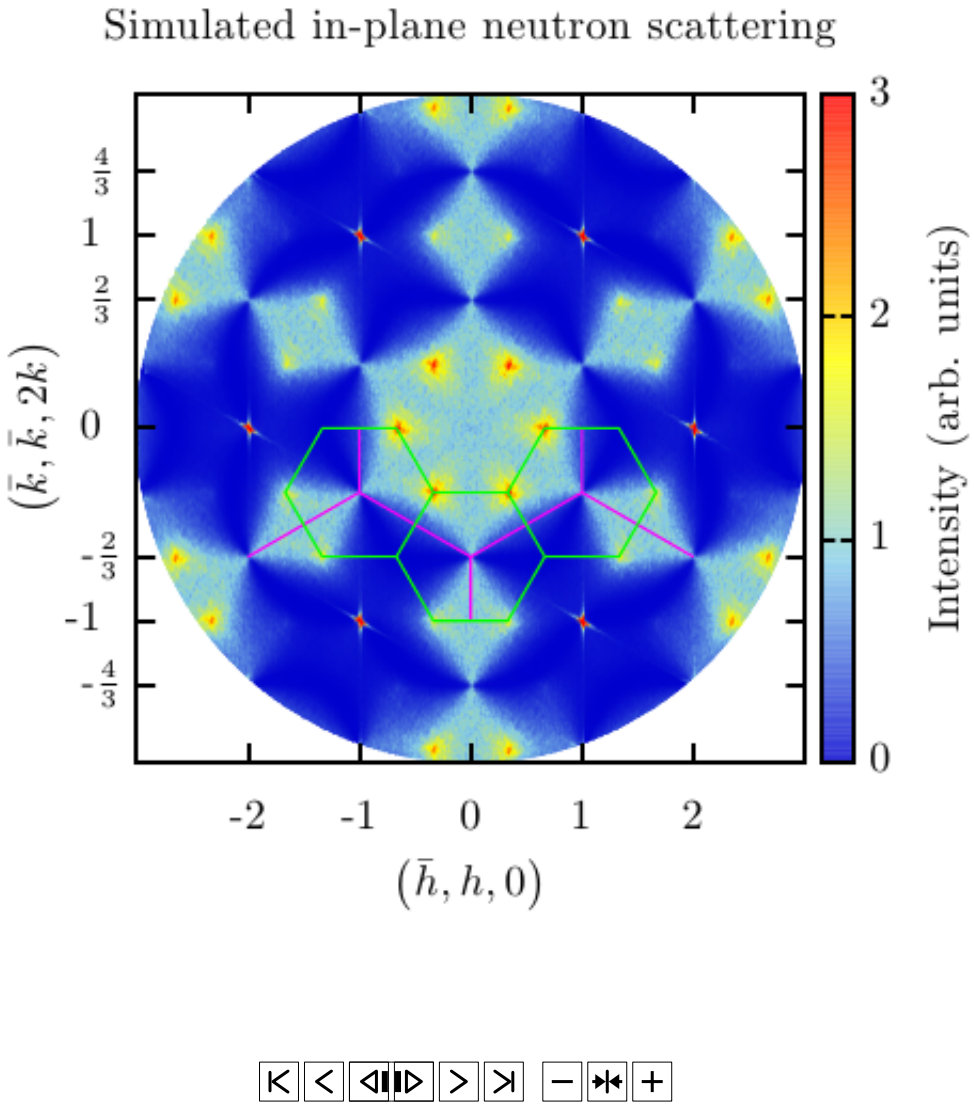

Figure 45: The evolution of the inplane component of the neutron scattering pattern of kagome ice in a field at an angle $\phi = 0°$. The thermodynamic variable is increased from $\frac{H}{T} = 0$ to $\frac{H}{T} > \frac{H_K}{T_K}$ to induce a Kasteleyn transition. In the electronic version of this document an animation is displayed whilst in the paper copy representative images from the animation are shown where, apart from at the high temperature starting point, the field was held constant.

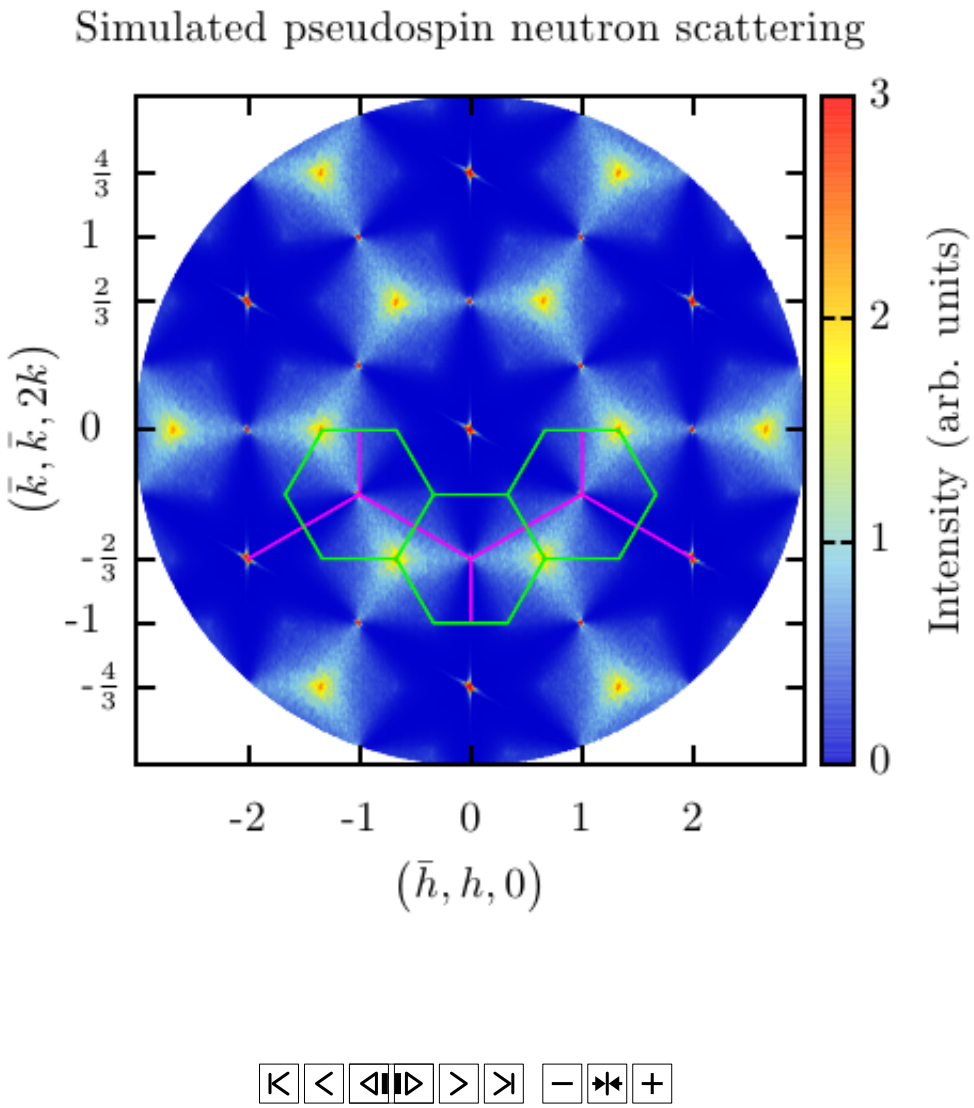

Figure 46: The evolution of the pseudospin component of the neutron scattering pattern of kagome ice in a field at an angle $\phi = 0°$. The thermodynamic variable is increased from $\frac{H}{T} = 0$ to $\frac{H}{T} > \frac{H_K}{T_K}$ to induce a Kasteleyn transition. In the electronic version of this document an animation is displayed whilst in the paper copy representative images from the animation are shown where, apart from at the high temperature starting point, the field was held constant.

the $[\bar{1}, \bar{1}, 2]$ direction which is consistent with the three-fold symmetry of the phase diagram. The scattering peaks drift continuously toward the Brillouin zone centres as predicted which facilitates the concentration of the scattering from a distribution over the scattering plane onto two major axes of intensity. Finally as the lattice approaches the long range ordered spin configuration the intensity of the scattering tends to a series of Bragg peaks on the high symmetry positions of the kagome reciprocal space lattice.

Examining cuts through the scattering maps at fixed values of either k or h illuminates the behaviour of the peaks more clearly as they increase or decrease in intensity or change position. The logarithmic peaks that are visible at the $h = \frac{1}{3}$, $k = \frac{1}{3}$ and five other symmetry related positions in the zerofield unpolarised scattering pattern, figure (39), are predicted to drift continuously toward the Brillouin zone centre as the system moves toward the Kasteleyn transition eventually being replaced by a Bragg peak once the long range ordered state has been achieved [33]. Figure (47) shows a series of linescans at $k = \frac{1}{3}$ where the field driving the system toward the transition is parallel to the y axis of the lattice and the movement of the logarithmic peaks is visible.

The intensity of the logarithmic peak is predicted to remain logarithmic as it drifts until the critical region just above the transition temperature where its behaviour changes and it builds on the peak position as a function of the applied field,

$$I_{Critical}(Bragg) \sim (H_K - H)^{\frac{1}{2}} \tag{209}$$

Once in the long range ordered state the intensity of the Bragg peaks scales linearly with system size as for a standard Bragg peak, see equation (207). Typically a Bragg peaks builds in intensity in a spherically symmetric manner however the broken symmetry created by a field in the kagome ice model causes the critical scattering to develop around the Bragg position according to the parallel and perpendicular correlation lengths $\xi_\perp$ and $\xi_\parallel$ such that the peak shape develops in an elongated form as shown in figure (49).

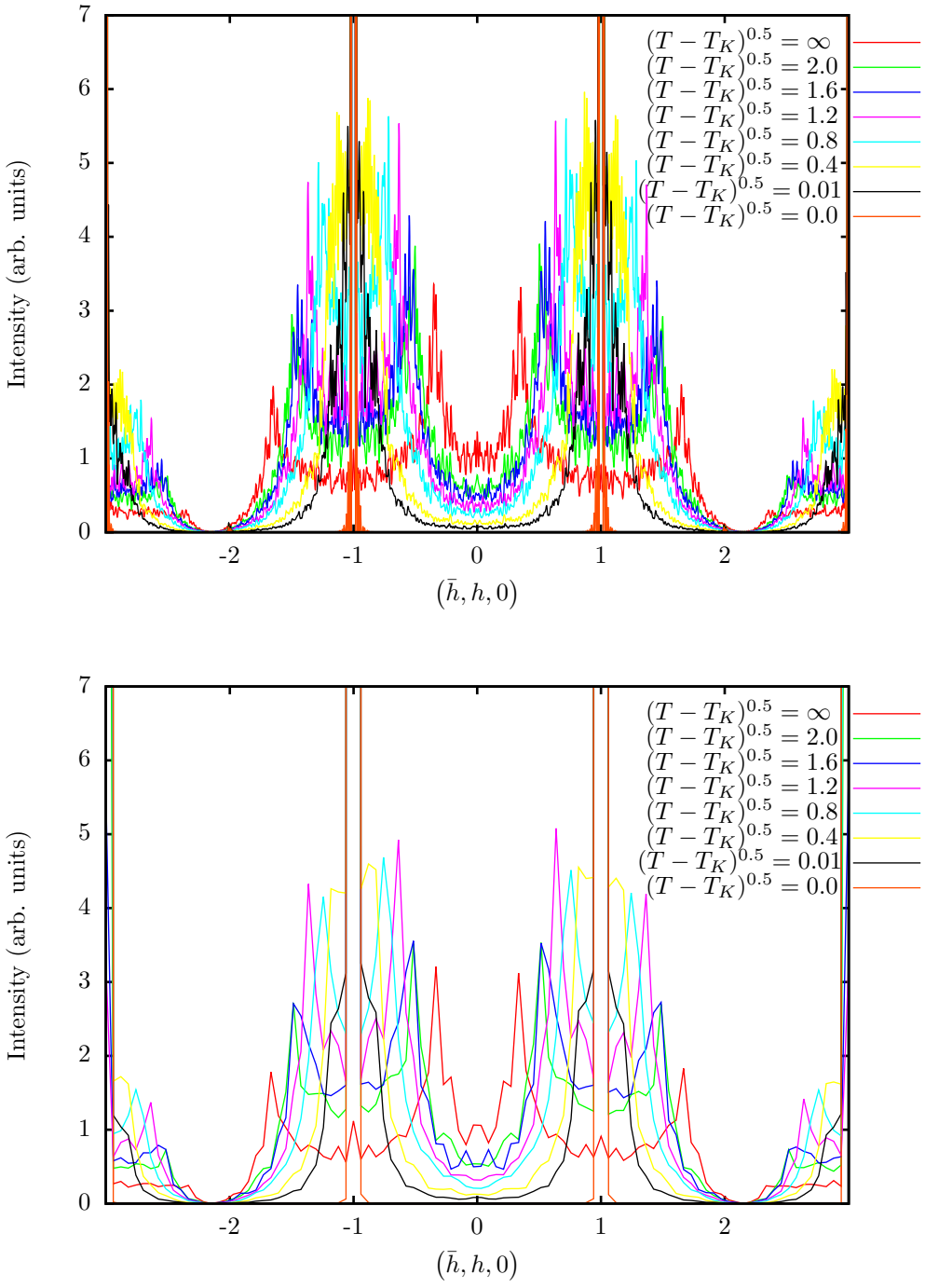

Figure 47: As the kagome ice lattice changes from a high temperature disordered regime to long range order at the Kasteleyn transition the position of the logarithmic scattering peaks drifts continuously towards the zone centres. Top: Raw data from the inplane component of the scattering taken at $k = \frac{1}{3}$ from the scattering shown in figure (45). Bottom: After applying a smoothing function to the data. The asymmetry of the peak intensities at $h = \pm\frac{1}{3}$ and $h = \pm\frac{5}{3}$ is clear and as the spin correlations develop toward the ordered state the peaks sharpen and move toward the zone centre as predicted.

6.1.4 Scattering Maps for $\phi > 0°$

The scenario in which the inplane field which induces the Kasteleyn transition is at an angle $0 < \phi \leq 60°$ to the $[\bar{1}, \bar{1}, 2]$ direction, see figure (35) for an illustration of ϕ, is now presented. When $\phi = 0°$ the high symmetry of the scattering pattern is reduced by the intensity building equally on the two high symmetry angles of the scattering map but when the field is rotated the intensity builds unequally on these angles favouring the angle closer to the field direction as the rotation increases. The scattering maps of all spin components also display the effects of the rotated field as a twisting effect in the scattering surrounding the logarithmic peaks which removes the two mirror axes from the scattering pattern in addition to the reduction from six-fold rotational symmetry to two-fold rotational symmetry as displayed in the $\phi = 0°$ case.

Figure (48) shows an animation (electronic document only) of the unpolarised spin component of the scattering for $\phi = 50°$ which shows the evolution of the scattering including the unequal weighting of the intensity and the twisted scattering around the peak positions discussed in the text. Simulations have also been performed at $\phi = 10°$, $20°$, $30°$ and $40°$ however the general progression of features is common to all simulations and so these are not displayed.

There are some crucial differences between patterns at different angles which are not due to a proportional progression with field angle. Figure (49) shows the intensity building on a Bragg position during the critical region for $\phi = 0°$, $10°$, $20°$, $30°$, $40°$, $50°$. For fields where the angle $0 < \phi < 60°$ the correlation lengths are predicted to be parallel and perpendicular to the field direction and the elongated peak shape to be rotated so that its long axis is perpendicular to the field, which can be seen to be qualitatively true in figure (48) but it is clear from figure (49) that this prediction is not completely satisfied, the long axis does not rotate continuously with the field and the extent of the correlations changes so that for $\phi = 30°$ the elongation is considerably reduced compared to $\phi = 0°$ or $\phi = 50°$, a phenomenon not accounted for by theory.

This can be explained as a consequence of the Ising character of the spins com-

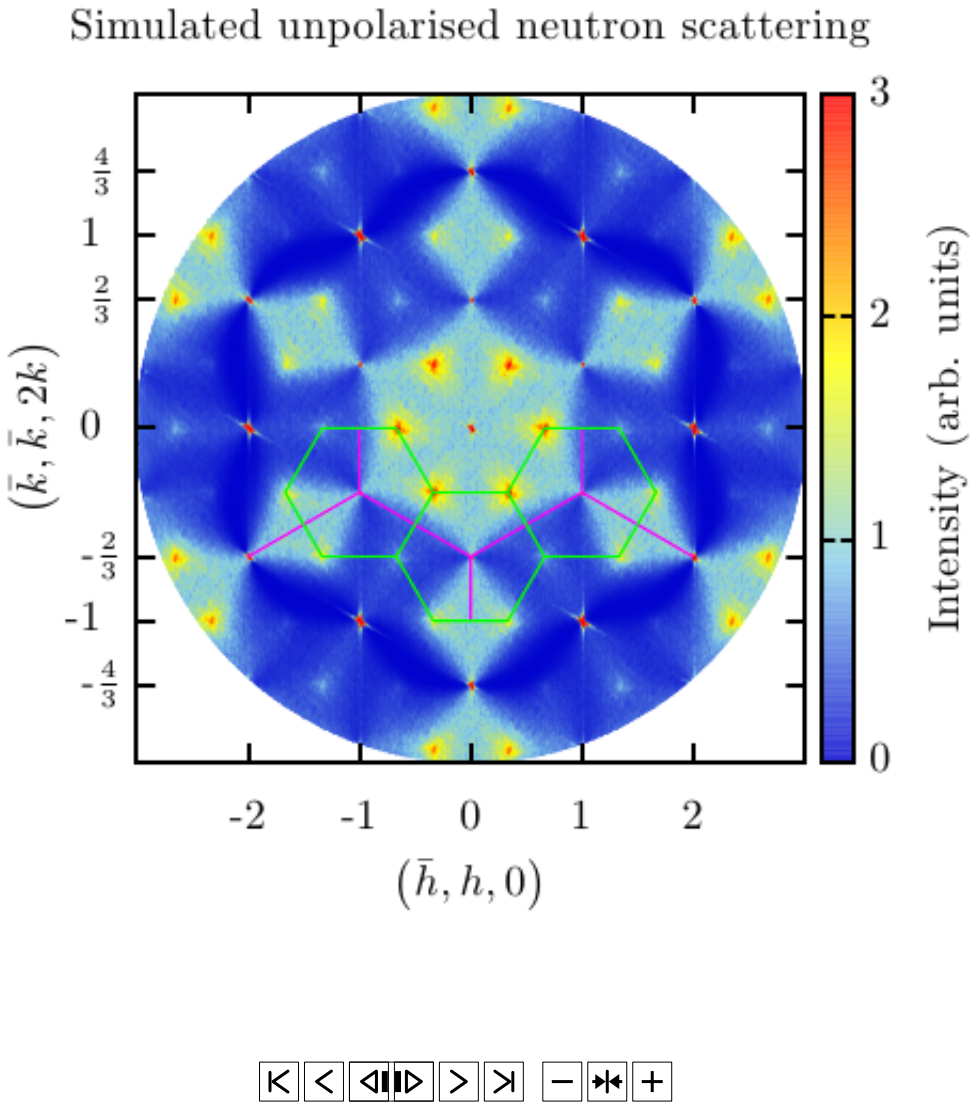

Figure 48: The evolution of the unpolarised component of the neutron scattering pattern of kagome ice in a field at an angle $\phi = 50°$. As the temperature is decreased toward the transition the six-fold rotational symmetry of the scattering pattern is reduced to a two-fold rotational symmetry and the mirror axes are removed. In the electronic version of this document an animation is displayed whilst in the paper copy representative images from the animation are presented.

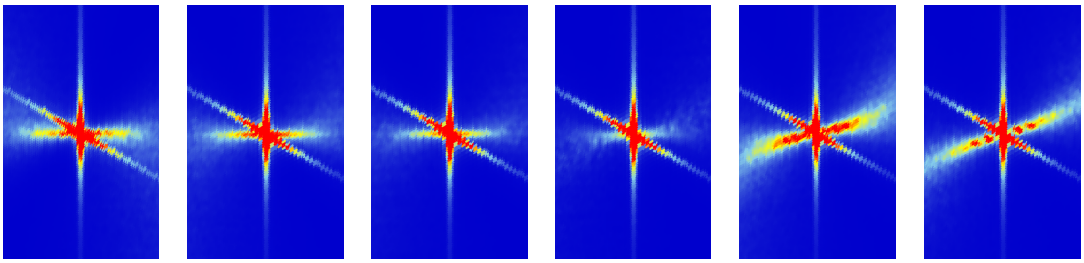

Figure 49: The peak shape of an intensifying Bragg peak in the critical region just above the transition temperature at approximately $\left(\frac{T}{T_K} - 1\right) = 1 \times 10^{-8}$, sharp scattering along the vertical and diagonal directions is a spurious effect due to the lattice shape. Typically intensity builds spherically about the Bragg position but the differing perpendicular correlation length critical exponents, $\nu_\perp = \frac{1}{2}, \nu_\parallel = 1$, cause the scattering to intensify in an elongated shape whose long axis is predicted to be perpendicular to the field direction. The behaviour of the critical Bragg peaks in the simulations shown here does not entirely match that prediction as explained further in the text.

bined with the lattice configuration and the loop formation process explained in chapter (5). Together these constraints mean that all spins are fixed to one of the three easy directions of the lattice hence they cannot align directly with the field unless it is parallel to one of these directions as this would require them to rotate away from their local Ising axis. Consideration of the lattice indicates that it is easy to develop correlations along the $\phi = 30°$ direction as the lattice naturally contains lines of spins in this direction. With field in this direction the loops of spins are 'commensurate' with the lattice such that those spins that are not part of the loops are left in an identical alignment, see for example the black spins in figure (55). In this case, although the perpendicular correlation length is theoretically predicted to diverge more slowly than the parallel one, the lattice configuration enforces the ordering of the perpendicular spins so that the correlation length in this direction is artificially extended and the critical peak shape reflects this with a considerably reduced elongation.

When the field is in a 'non-commensurate' direction such as $\phi = 0°$, a correlation develops in a direction approximately parallel to the field through the creation of loops with a ballistic trajectory along the field direction and a stochastic element perpendicular to it. In the case $\phi = 0°$ this means that loops will correlate all spins

labelled 1 but will randomly flip an approximately equal number of spins labelled 2 or 3 so that the correlation parallel to the field is strong but the correlation perpendicular to it is weak giving the observed elongated scattering peaks. Loops with ballistic trajectories parallel to $\phi = 0°, 30°$ and $\phi = 60°$ provide the dominant paths through the lattice and the loops formed by fields at other field angles can be seen as perturbed versions of these three trajectories rather than loops forming in a separate way. As a consequence of the loops jumping between these easy paths the critical peak shape does not rotate continuously but appears to have its long axis parallel to the x axis until $\phi > 30°$ after which it lies at an angle of approximately $30°$ to the x axis. When the long axis of the scattering changes from $\phi = 0°$ to $\phi = 30°$ the diffuse scattering is smeared out, this is particularly visible at $\phi = 30°$ and $\phi = 40°$, along both of the competing directions which gives the impression of rotation around the central point of each Bragg peak.

The behaviour of the spin correlations within the kagome ice model is more complex than predicted by theory as they are influenced by the discrete lattice and spin directions as well as the parallel and perpendicular correlation length critical exponents. Extending a finite lattice to a theoretical model of infinite extent results in the discrete directions tending toward a continuous spectrum so that it should be possible to recover behaviour controlled solely by the critical exponents and observe a smooth rotation of the critical peak shape. More experimental work is required to determine whether it is possible to see such finite size effects in kagome ice materials.

6.1.5 Comparison with Experiment

The first experiment investigating this behaviour in kagome ice has been carried out in response to the work presented here. Tom Fennell has taken neutron scattering measurements of a single crystal of the spin ice material $Ho_2Ti_2O_7$ in the presence of a magnetic field with a small tilt away from the $[1, 1, 1]$ direction at the Institute Laue-Langevin. The aim of his work was to record the neutron scattering patterns of a kagome ice material near to and at a Kasteleyn transition and the data recorded there is directly comparable to the simulations produced in this work as

the pseudospin structure factor should describe what is observable experimentally with polarised neutrons and the complete spin structure factor should describe what is observable with unpolarised neutrons.

Experimentally the Kasteleyn transition would ideally be effected by aligning the external field along the $[1,1,1]$ direction and then tilting it toward the $[\bar{1},\bar{1},2]$ direction thus introducing a component of the field onto the kagome plane at $\phi = 0°$ which would drive the system through the transition at the maximum transition temperature, see figure (50) for the relation between the experimental field angle and the inplane field angle ϕ used in this work. In reality it is difficult to control the precise field alignment with respect to the crystal and the interaction between the field and the crystal further perturbs the field direction within the crystal [58] so that the applied field will almost certainly have $\phi \neq 0°$. As the experimental field is measured with respect to the $[1,1,1]$ axis then small deviations create large changes in the field components on the kagome plane so that accuracy of alignment to within a degree, which is generally suitable for experiments on magnetic materials, still creates large uncertainties in this experimental configuration. This makes this system particularly interesting and challenging to study as it is unusual for very small changes in field direction to cause such large experimental responses.

Figure (51) shows a scattering map where the experimental field was measured as a tilt away from the $[1,1,1]$ direction of approximately 3° along the $[\bar{1},\bar{1},2]$ direction and approximately 1° along the $[1,\bar{1},0]$ direction but error in the recorded field angles means the inplane field angle could easily have been in the range $15° \leq \phi \leq 25°$. This is presented alongside simulated data at $\phi = 20°$ which gives a good fit to the experimental data. We note here that quantitatively matching simulated to experimental data would provide a method of measuring the inplane field angle more accurately than is currently possible in experimental situations. See also figure (19) for an additional image of the kagome ice neutron scattering patterns.

The experimental data was recorded at a temperature of 0.1 K and a field of 0.4 T. In order to compare these values with the theoretical field and temperature used in the simulations it is necessary to include a value of the moment per spin for

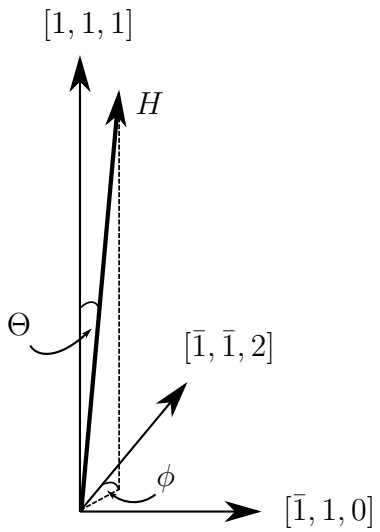

Figure 50: During the experiments performed in reference [67] the tilt angle between the field, H, and the $[1,1,1]$ direction is marked in the figure and was measured as $\Theta \approx 3°$. Only the component of the field that lies in the kagome plane (perpendicular to $[1,1,1]$) causes the Kasteleyn transition hence this is the field referred to in this work. Note that a very small tilt of the experimental magnetic field can produce a large angle on the kagome plane and cause the spin correlations to change significantly.

$Ho_2Ti_2O_7$ which is approximately $\mu = 10\ \mu_B$ and resolve the experimental field into the kagome plane component

$$\frac{H_{th}}{T_{th}} = \frac{10\mu_B H_{exp} \sin 3°}{k_B T_{exp}} \tag{210}$$

$$\implies \frac{H_{th}}{T_{th}} = 1.406\ldots \tag{211}$$

whilst the simulation was performed at,

$$\frac{H_{th}}{T_{th}} = \frac{1}{7.7751\ldots} = 0.128\ldots \tag{212}$$

and the Kasteleyn transition is predicted to occur, equation (147), at

$$\frac{H_{th}}{T_{th}} = 0.264\ldots \tag{213}$$

The difference between the simulated and the experimental thermodynamic variable is large however this is to be expected as there are multiple reasons why the experimental system may not agree with the theory at these temperatures and fields.

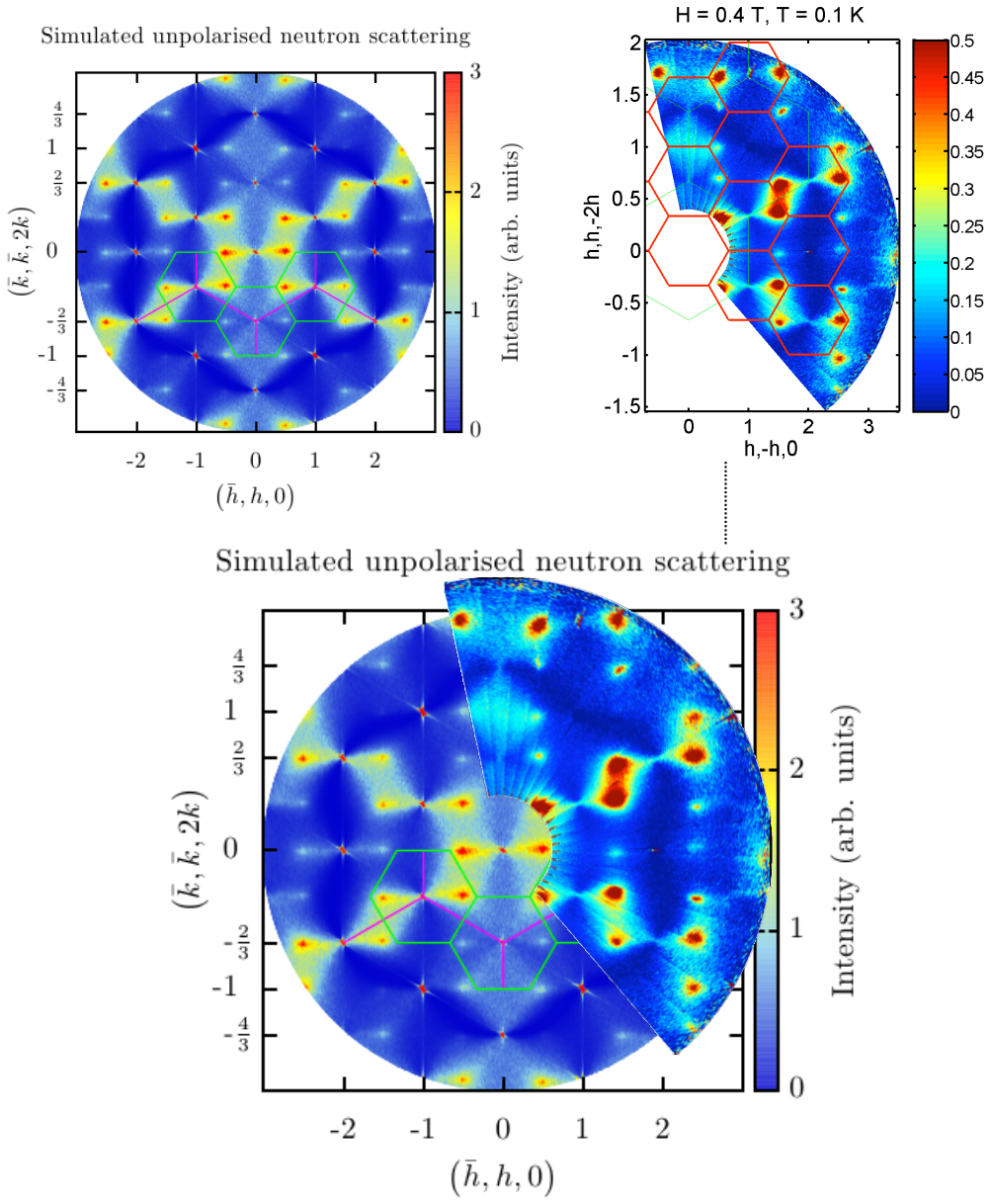

Figure 51: Top left: Unpolarised neutron scattering map at $\frac{T}{T_K} = 2.06$ with $\phi = 20°$. Top right: Experimental results under similar conditions, further detail in the text. Centre: Experimental results overlaid on simulated data showing the good level of agreement between the two. The intensity is scaled differently in the experimental data producing larger regions that appear to have very high intensity on the peak positions (larger red areas) however the match between the shape and angle of the features in the diffuse scattering is very good.

Demagnetisation effects may alter the field experienced within the sample and as discussed previously a small angular change will have significant effects on this system. For example demagnetisation effects would only need to reduce the effective field angle from around three degrees to $\phi = 0.273°$ in order for the previous calculation to produce exactly matching thermodynamic parameters. At an experimental temperature of $T = 0.1$ K the energy scale of the predicted dipolar groundstate [35] becomes significant and there is likely to be competition between this and the Kasteleyn groundstate as well as a general loss of ergodicity which may introduce dynamic restrictions to the system.

The simulation data shown in figure (51) was produced at $\frac{T}{T_K} = 2.06$ and matches the experimental data very closely. As the temperature is reduced toward $T = T_K$ the simulated scattering map evolves to display only Bragg scattering, cf. figure (48). Experimentally the data does not evolve in the same way and the correspondence with the simulation breaks down at approximately $\frac{T}{T_K} = 2$. A combination of demagnetisation effects, groundstate competition and ergodicity problems conspires so that it has so far been impossible to observe the Kasteleyn transition in $Ho_2Ti_2O_7$ however this remains an ongoing experimental target.

6.2 Thermodynamics

Further information regarding the Kasteleyn transition may be gained by investigating the thermodynamic simulation data for the kagome ice model.

6.2.1 Biaxial Magnetisation

The magnetisation of the kagome ice model discussed here is always the component of the total magnetisation that lies in the kagome plane only. The choice of a pinning field along one of the $\langle 1, 1, 1 \rangle$ directions produces a permanent component in that direction which is uninteresting in this context hence it is disregarded. The magnetisation of the kagome ice model is of course a measure of the spin arrangement and the asymmetric nature of the Kasteleyn transition is well reflected in this quantity. Each of the three long range ordered states of the system below the transition is

a vacuum for fluctuations that would alter the spin configuration and all up triangles have the same two spins pointing in, one spin pointing out configuration with identical choice of the out spin according to which of the three continuous topological sectors is dominant. Measured with respect to an axis parallel to the ordering vector for the topological sector this produces a magnetisation with an unchanging maximum value at all temperatures below the transition. Geometric considerations indicate that the projection of each spin onto the kagome lattice is $|\mathbf{S}_i| = \frac{2\sqrt{2}}{3}$ and in the ordered state the maximum total spin projection along the ordering axis for each triangle is equivalent to two of the three spins, see figure (31).

$$M_{max} = \frac{2}{3} \times \frac{2\sqrt{2}}{3} = 0.6285\ldots \tag{214}$$

Above the Kasteleyn transition strings of spins are able to flip altering the magnetisation of the lattice. In the high temperature or zero field state the spin configuration is an approximately equal weighting of the three long range ordered states and so the magnetisation tends to zero in this limit.

Figure (52) shows the magnetisation as a function of temperature and illustrates its deviation from saturation with a power-law dependence on the thermodynamic variable (cf. the Landau and thermodynamic analyses presented in sections (1.3.3) and (4.5)),

$$(M_{max} - M) \sim \left(\frac{H_K}{T_K} - \frac{H}{T}\right)^{\frac{1}{2}} \tag{215}$$

The correct order parameter in this system is $(M_{max} - M)$ rather than simply the magnetisation in order to provide a positive, increasing order parameter, however it is unusual that the most naturally defined order parameter is zero in the more ordered state and grows as the disordered state is entered.

The theoretical behaviour of the magnetisation was calculated in chapter (4), equations (174) and (175), and this is clearly well matched by the simulation. This magnetisation curve is specific to the topological sector and any deviations from it would necessarily indicate a violation of topological constraint. Modelling kagome ice with a single spin flip algorithm is not appropriate as it would generate topo-

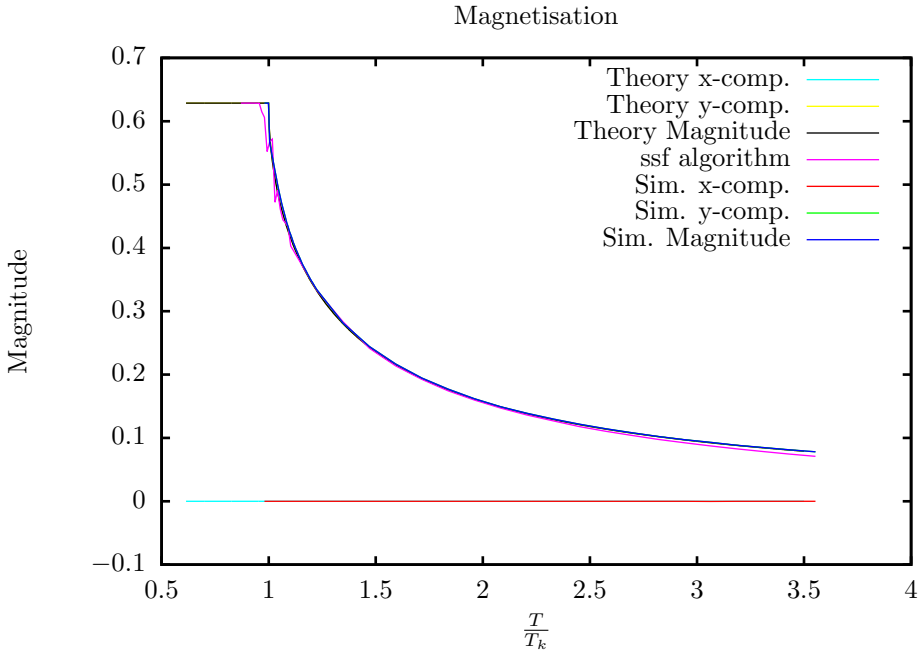

Figure 52: Magnetisation of the kagome ice lattice as a function of temperature. Below the transition the magnetisation is constant at $M = M_{max} = \frac{4\sqrt{2}}{9}$ and above it decreases to zero with a power-law dependence as described in the text. The magnitude of the vector magnetisation is shown alongside the theoretical magnetisation, see equations (174) and (175), and there is excellent agreement between the two throughout the simulation. For comparison the same simulation is shown when a single spin flip update algorithm is used instead of the loop updates. Topological defects generated by the single spin flip algorithm violate the local constraint in the kagome ice model and are predicted to produce rounding of the sharp transition where the energetic cost of a single spin flip is similar to the energy scale of the lattice. At temperatures $T > 1.3T_K$ these violations do not cause a significant difference between the predicted and simulated magnetisation but near to the transition the simulated magnetisation shows evidence of rounding and increased noise as expected.

logical defects within the lattice and the loop algorithm described in chapter (5) is required if the simulation is to match the theory, see figure (52), particularly in the region near to the transition. The theoretical equations and figure (34) also indicate that the magnetisation can take any direction within the kagome plane and can be decomposed into a y-component (defined parallel to the $[\bar{1}, \bar{1}, 2]$ direction) and an x-component (defined parallel to the $[1, \bar{1}, 0]$ direction).

In the case when the applied field direction is no longer parallel to the moment of the long range ordered state reached at the Kasteleyn transition, $\phi > 0°$, the system displays a biaxial magnetisation; near to the transition the magnetisation of the lattice is no longer parallel to the applied field direction. At high temperatures when the lattice is disordered it is able to accommodate strings and loops of spins of all types and directions so that the magnetisation of a large system is able to match the direction of the applied field increasingly accurately as the thermodynamic limit is approached. The magnetisation of the lattice can only be altered by string excitations and a lattice containing $N = 3L^2$ spins can hold a maximum of L strings. By varying the number of strings and their path across the lattice the system is able to take on a discrete set of overall magnetisation directions and as the lattice size increases these discrete magnetisation angles tend towards a continuous spectrum hence the ability of the lattice to develop a magnetisation exactly parallel to the field direction increases with system size.

As the temperature is decreased and the Kasteleyn transition is approached the magnetisation builds in the direction of the field, however for all field angles $-60° \leq \phi \leq 60°$ the long range ordered state reached at the transition is the same and has a magnetisation parallel to the $[\bar{1}, \bar{1}, 2]$ or $\phi = 0°$ direction, see figure (31). As the increasing field or decreasing temperature drives the system toward the Kasteleyn transition the lattice configuration must therefore rearrange itself during the critical regime just above the transition in order to reach the required long range ordered state and in doing so the x-component of the magnetisation must be reduced to zero. From this point and throughout the long range ordered state the lattice exhibits a biaxial magnetisation in a different direction to the applied field.

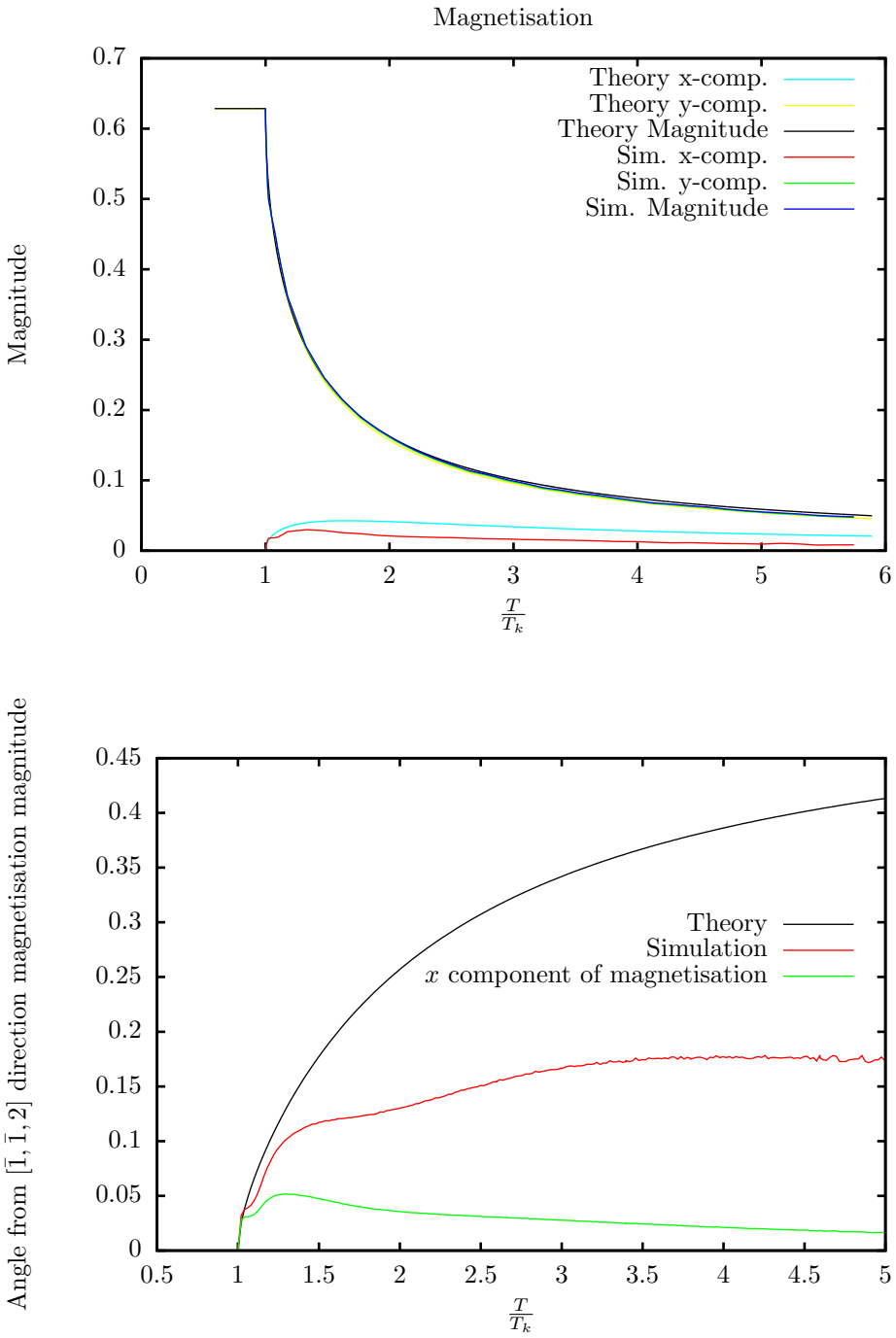

Figure 53: Top: The complete behaviour of the magnetisation with a field at an angle $\phi = 30°$ showing simulation and theoretical results. As the temperature is decreased the lattice begins to develop a finite component of the magnetisation parallel to the x axis due to the influence of the magnetic field but as the ordered state reached at the transition has zero x-component magnetisation this is removed by lattice re-arrangement in the critical region just above the transition. Bottom: The vector magnetisation subtends an angle to the $[\bar{1},\bar{1},2]$ direction or y axis due to the influence of the magnetic field however as the spin configuration evolves toward the long range ordered state this angle decreases to zero with the power law behaviour exhibited throughout the model. The x component of the magnetisation in the top image is reproduced at a larger scale to highlight its behaviour. Here the theoretical subtended angle indicates that by $T \approx 5T_K$ the vector magnetisation should be quite well aligned with the applied field direction however the simulated vector magnetisation does not evolve in this manner and never approaches the field angle.

Figure (53) shows the magnetisation for $\phi = 30°$ and the increasing x component is clearly visible as the transition is approached from the high temperature side indicating the magnetisation is increasing in the direction of the applied field. The lower panel of this figure shows the angle between the $[\bar{1}, \bar{1}, 2]$ direction and the vector magnetisation which illustrates the decrease to zero at the transition governed by a power law as with all the critical quantities in the system. The theoretical angle subtended by the vector magnetisation increases toward its limiting value where it is parallel to the applied field direction following the expected behaviour for a system in the thermodynamic limit as described above but the simulated x-component of the magnetisation does not match the theoretical prediction and quickly falls below it reaching a maximum significantly less than the limiting value. This is almost certainly a numerical artefact of the simulations due to the finite size of the lattice which, although large, $N = 10800$ spins, is not able to take on a continuous range of magnetisation directions. Both the x component of the magnetisation and the angle the magnetisation vector subtends indicate that the behaviour the lattice exhibits is a complicated function of the allowed spin configurations combined with the relative weights of the probabilities controlling the loop algorithm at any temperature. The theoretical predictions are made in the thermodynamic limit and, particularly for $\phi \neq 0°$, do not completely match the simulation results.

6.2.2 Theoretical Finite Size Scaling Behaviour

The discrepancy between the theoretical predictions for the thermodynamic variables such as the x component of the magnetisation and the simulation results indicate that finite size effects, and shape effects, are likely to make a significant contribution to the behaviour of the kagome ice model. A finite size scaling analysis of this region should also validate the separate values of the critical exponent for the correlation lengths parallel and perpendicular to the applied field direction.

Bhattacharjee and Nagle have investigated the finite size scaling response of a model of dimers on a brick lattice which has been shown to be equivalent to the kagome ice model [115]. In that work they investigated the consequence of reducing

the extents of the lattice in turn from the thermodynamic limit, $\infty \times \infty$, to either $N \times \infty$ or $\infty \times M$ and finally to a finite $N \times M$ lattice. In this work we maintain a constant lattice shape of $L \times L$ with periodic boundary conditions hence we cannot see effects due to an actual changing lattice shape however we shall show that applying an inplane magnetic field to the kagome ice model has a similar effect to changing the shape of the lattice. By considering the specific heat they show that finite size scaling theory is appropriate to the critical region of the model and hence also to kagome ice. It was necessary to confirm this as both lattice models are subject to a topological constraint which imposes a violently asymmetric transition on the system and finite size scaling theory is generally applied to isotropic models with symmetric transitions thus it was not immediately obvious that the procedures could be carried from one situation to the other.

Bhattacharjee and Nagle prove that the specific heat finite size scaling function depends on the scaled reduced temperature and a shape factor, $R = \frac{N^2}{M}$ where there are $2N$ lattice points along the horizontal direction and $2M$ lattice points along the vertical direction of the brick lattice. The scaled reduced temperature is defined as $\tau = \frac{MN^2 t}{M+N^2}$ where the reduced temperature is $t = \frac{T-T_K}{T_K}$ as usual. The specific heat scaling function may be written in terms of these variables in the critical region.

$$C(T) \sim \mathcal{P}(R, \tau) \tag{216}$$

The behaviour of this scaling function as a function of τ varies from the limit $R \to 0$ where it tends to a series of delta functions to the opposite limit $R \to \infty$ where it becomes a smoothly varying single peaked function as shown in figure (54). In the two cases where Bhattacharjee and Nagle investigate a semi-infinite lattice they consider the limiting behaviour as the finite extent of the lattice also tends toward infinity, $\infty \times (M \to \infty)$ for example, and find the behaviour remains different at the two limits, although they are then both for a lattice of the form $\infty \times \infty$. The general $\infty \times \infty$ lattice must therefore have the ability to display a continuous range of behaviour for $0 < R < \infty$ depending on a factor governing an effective shape within the lattice.

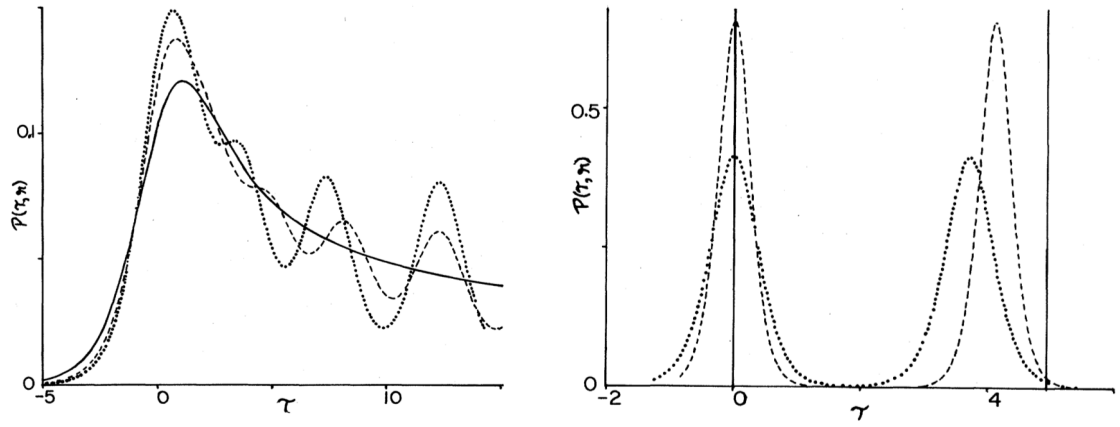

Figure 54: The behaviour of the finite size scaling function for the specific heat of the brick lattice model as a function of scaled reduced temperature depends on the shape factor, R. The lattice contains $2N$ lattice points in the horizontal direction and $2M$ lattice points in the vertical direction and the shape factor is defined as $R = \frac{N^2}{M}$. Left: $R > 1$ The limit $R = \infty$ is shown with a solid line, $R = 3$ is dotted and $R = 5$ is dashed. As $R \to \infty$ the first peak of the function tends to the peak of the limiting function and the oscillatory peaks are reduced to a smooth line. Right: $R < 1$. The limit $R = 0$ is a series of delta functions shown with solid lines, $R = \frac{1}{3}$ is dotted and $R = \frac{1}{5}$ is dashed. The position of the first peak remains fixed whilst the subsequent peaks shift toward their limiting positions as the vertical extent of the lattice increases. Figure copied from reference [115].

The Limiting Case $R = 0$ On the brick lattice this shape factor corresponds to an infinitely tall and thin lattice and the specific heat becomes a series of delta functions, see figure (54), right. The extent of the lattice in the horizontal and vertical directions may be varied to produce the limit investigated here but identical periodic boundary conditions are enforced on the lattice for all shape factors creating a toroidal topology so that the impact of the shape factor on the properties of the model is separate from that of the lattice topology.

The first delta function in the specific heat is exactly at the transition temperature and corresponds to the first string being placed into the lattice. As discussed in section (4.4) in order to place the second string into the lattice the temperature must increase by a small amount to compensate for the increased unfavourable Zeeman energy associated with each string hence there is a gap in the specific heat then, once the temperature has increased by the necessary amount, it becomes favourable to add a second string to the lattice and the specific heat exhibits a second delta function. Between these temperatures the system is frozen in a similar way to the long

range ordered ground state configuration but with the addition of a single string. The positions of these delta functions scale with the system size and in the thermodynamic limit they all tend toward the bulk critical temperature. The lattice shape restricts the path the forming string may take across the lattice so that in the $R = 0$ limit there is only one possible path and even when observed as a thermal average of energy fluctuations (such as the specific heat) this creates a series of delta functions as subsequent strings are added to the lattice. Each integrated delta function (over temperature) is proportional to the energy required to place that string into the lattice [103].

The Limiting Case $R = \infty$ This shape factor corresponds to an infinitely short and wide brick lattice and the specific heat appears as a smooth curve with a single maximum at a temperature slightly greater than the transition temperature, see figure (54). The energy associated with a forming string is proportional to its magnetisation and as the extent of the lattice in the direction of the forming string tends to zero the change in energy or magnetisation upon adding that string also tends to zero. This allows the strings to enter the lattice continuously such that the specific heat is a smooth curve. Figure (54) shows that whilst the signature of each additional string being placed into the lattice is most striking for $R = 0$ it remains present as oscillations in the specific heat for all lattice shapes until $R = \infty$ when the extent of the lattice in the direction perpendicular to the string formation becomes infinite.

6.2.3 Topological Winding Numbers

The behaviour of the specific heat of the brick lattice model is a function of the shape factor for the lattice however this quantity is constant for all simulations carried out on the kagome ice lattice although it will be shown that the specific heat (and magnetic susceptibility) vary as if it were changing. A relevant quantity to examine instead is a measure of the topological order of the lattice for the addition of the first string at the Kasteleyn transition using a topological quantum number. This quantity is often known as a winding number and for a closed loop and a point

in a 2D plane it is generally defined as the number of times the loop winds around the point in the anticlockwise direction. In kagome ice periodic boundary conditions transform a plane into a torus and similarly the winding number is transformed into the number of times a loop closing through the boundary conditions winds around a line on the torus.

The winding number, W_i, for a topological state C can be calculated by taking a cut through the lattice in the direction of a lattice vector i and recording a cut number, $Y_i(C)$ which is increased by one for each string it cuts [38]. The winding number is then defined as the difference between the cut number in a particular state and that of a reference state, C'. The first loop to transform the system out of the long range ordered state at the Kasteleyn transition must have a vertical component due to the lattice configuration hence these loops are of interest for this problem and the appropriate lattice vector is a in the horizontal direction; however, by taking a cut parallel to the lattice vector b it is equally possible to define a winding number that measures horizontal loops. During this derivation these will not be considered and the winding number is referred to as $W_a = W$.

$$W = Y_a(C) - Y_a(C') \tag{217}$$

where the reference state is chosen to have a single string crossing the lattice once so that the winding number records the difference between the number of times a loop winds around the lattice and the minimum number of times it can wind around the lattice.

The winding number records the number of times a loop is intersected on the lattice thus the parameters governing the behaviour of the loop are particularly important. Chapters (4) and (5) describing the Kasteleyn transition and the loop algorithm indicate that the path of a forming string is influenced by the probability to choose a particular spin at each up triangle and an overall attempt to minimise its projection onto the applied field. As the inplane field angle, ϕ, is increased the angle of the string changes accordingly, continually attempting to minimise its Zeeman energy interaction as shown in figure (55). It is helpful to consider the

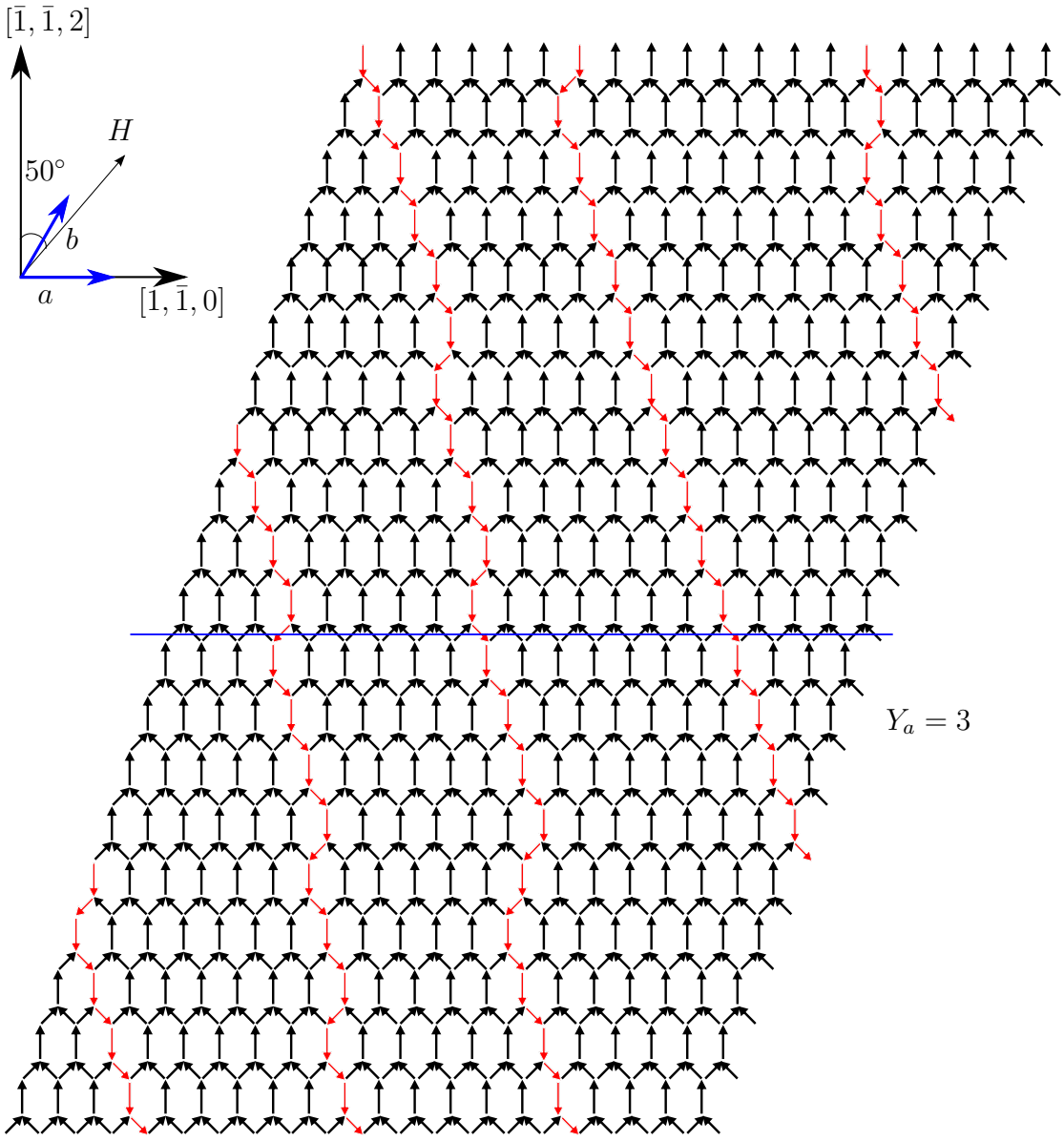

Figure 55: A snapshot of the kagome ice lattice taken from a simulation at $T = T_K$ and $\phi = 50°$. The lattice contains a single string (highlighted in red) which winds around the lattice through the periodic boundary conditions until it closes on itself. This snapshot shows the loop forming in a direction that is approximately perpendicular to the field direction as this minimises its unfavourable Zeeman interaction. The blue line parallel to the lattice vector a is a trajectory for calculating a cut number which records the number of times a string is encountered in that direction.

path of a forming string as being composed of a ballistic component combined with a perpendicular stochastic component due to the entropic forces inherent within the Gibbs ensemble which can be biased due to the applied field but remains non-deterministic. A string will preferentially form with its ballistic component parallel to one of the 'easy' paths across the lattice which are at an angle $\phi = 0°$, $30°$ or $60°$, cf. figure (49), but the stochastic component will be biased to introduce a perpendicular drift to these directions so that on average the string forms an angle with the y axis which will increase with increasing field angle, then as the string reaches the boundary of the lattice it re-enters on the opposite side such that overall it winds around the lattice.

When placing the first string into the lattice at the Kasteleyn transition temperature the constraints of the topologically ordered state mean that for field in the $[\bar{1}, \bar{1}, 2]$ direction the string must form parallel to the field direction and hence have a large Zeeman energy cost. As the field angle increases the string is able to find paths that have a smaller projection onto the field producing a smaller Zeeman energy cost and therefore reducing the temperature at which a string may be placed into the lattice, see figure (32). The topology of the lattice means that when the field angle has reached $\phi = 60°$ it is possible to place a string into the lattice that is perpendicular to the field direction and therefore has no Zeeman energy cost. I suggest that this analysis provides a particularly straightforward explanation for the observed maximum of the transition temperature, see figure (32), at $\phi = 0°$ falling to $T_K = 0$ at $\phi = 60°$.

Using this terminology if the field is applied along the y axis the strings will form with a ballistic component parallel to that direction and a stochastic component with no bias with respect to the $\pm x$ direction so that the average angle between the string and the y axis is zero and the string finds its origin and becomes a loop after crossing the lattice only once. The winding number is therefore,

$$W = Y_a(C) - Y_a(C') \tag{218}$$

$$= 1 - 1 = 0 \tag{219}$$

The lattice construction dictates that when the field angle $\phi = 60°$ the string will be able to form such that it crosses the lattice only once but along a lattice diagonal rather than vertical direction. As this produces a loop which winds around the lattice only once it also has a winding number, $W = 0$. If the field is applied at any other angle then when the string that starts to form across the lattice crosses the periodic boundaries along the top and bottom of the lattice it will not enter collinear to its initial trajectory and will wind around the lattice more than once before closing to become a loop with $W > 1$. The extent to which a string winds around the lattice is a complex decreasing function of the applied field and the lattice geometry combined with the periodic boundary conditions. If ϕ is only slightly larger than zero then the first string will wind many times around the lattice before closing as its drift along the x-axis will be small and so the winding number will be large. If ϕ is increased further the first string forms at a larger angle to the y-axis to minimise its Zeeman interaction and so winds around the lattice fewer times before closing and as ϕ approaches 60° more closely the string is able to find an easier path increasingly parallel to the lattice diagonal direction and the winding number decreases.

In order to categorise this behaviour I introduce the concept of a commensurate string. Such a string only winds around the lattice once as the combination of field direction, loop algorithm probabilities, lattice geometry and periodic boundary conditions are commensurate. In contrast, changes in any one of these factors will produce an incommensurate string that winds around the lattice at least twice and at most L times for a lattice of $3L^2$ spins.

The commensurability of a string with the lattice, Γ, is inversely proportional to the winding number and directly proportional to the shape factor of reference [115],

$$\Gamma = \frac{1}{W} \propto R \tag{220}$$

The behaviour of the kagome ice model under the influence of a magnetic field can now be compared to that of the brick lattice with varying lattice shapes and remarkably the influence of the applied magnetic field direction is equivalent to varying the extent of the lattice.

6.2.4 Kagome Ice Finite Size Scaling Behaviour

The thermodynamic variable in kagome ice is the ratio $\frac{H}{T}$ rather than either of these parameters separately so that the specific heat and the susceptibility are related by

$$\frac{C}{\chi} = \frac{2H^2}{T} \tag{221}$$

and behave in the same manner. The susceptibility of kagome ice is considered here however this still permits a direct comparison with reference [115].

The magnetic susceptibility is inversely proportional to the reduced temperature to the power of the critical exponent γ as shown in table (2),

$$\chi \sim |t|^{-\gamma} \tag{222}$$

and similarly the correlation lengths in different directions are proportional to the reduced temperature to the power of exponents corresponding to those directions,

$$\xi_{\perp} \sim |t|^{-\nu_{\perp}} \tag{223}$$

$$\xi_{\parallel} \sim |t|^{-\nu_{\parallel}} \tag{224}$$

where for the directions perpendicular and parallel to the applied field

$$\nu_{\perp} = \frac{1}{2} \tag{225}$$

$$\nu_{\parallel} = 1 \tag{226}$$

which were shown to be correct through explicitly calculating the dimer pair correlation functions [116] and a renormalisation group approach where the exponents were shown to be independent of dimension [117]. Further verification for these values is provided by their agreement with Fisher's anisotropic hyperscaling relation [118],

$$(d-1)\nu_{\perp} + \nu_{\parallel} = 2 - \alpha \tag{227}$$

where

$$d = 2 \tag{228}$$

$$\alpha = \frac{1}{2} \tag{229}$$

The finite scaling response of the susceptibility depends on the commensurability of the strings just as the specific heat of the brick lattice model depended on the shape function and the limiting commensurate and incommensurate behaviour is now presented.

Commensurate Field Behaviour The scaling hypothesis for the specific heat of the brick lattice model is dependent on a shape function and the scaled reduced temperature, equation (216), but as the kagome lattice is a constant shape and initially we choose to set the applied field at an angle $\phi = 0°$ the shape function or commensurability is constant and may be discarded leaving behaviour that is expected to be controlled by the parallel and perpendicular correlation lengths only. A scaling hypothesis can be written which encapsulates the critical divergence in terms of these variables,

$$\chi = t^{-\gamma}\mathcal{P}\left(\frac{\xi_\perp}{L_\perp}, \frac{\xi_\parallel}{L_\parallel}\right) \tag{230}$$

where the correlation lengths are scaled by the length of the lattice edge as this is the largest length in the system and the only fixed reference in the critical region. Here L is a numerical parameter determining the lattice length which is normalised by the lattice constant and is therefore dimensionless and the shape of the lattice fixes $L_\perp = L_\parallel = L$. Substituting for the correlation length using equations (223) and (224),

$$\chi = t^{-\gamma}\mathcal{P}\left(\frac{1}{|t|^{\nu_\perp}L}, \frac{1}{|t|^{\nu_\parallel}L}\right) \tag{231}$$

$$= t^{-\gamma} \times (|t|^{\nu_\parallel}L)^{\frac{\gamma}{\nu_\parallel}}\,\mathcal{Q}\left(\frac{1}{|t|^{\nu_\perp}L}, \frac{1}{|t|^{\nu_\parallel}L}\right) \tag{232}$$

$$= L^{\frac{\gamma}{\nu_\parallel}}\mathcal{Q}\left(\frac{1}{|t|^{\nu_\perp}L}, \frac{1}{|t|^{\nu_\parallel}L}\right) \tag{233}$$

$\phi = 0°$ and in the limit $|t| \to 0$, $|t|^{\nu_\perp} \gg |t|^{\nu_\parallel}$. It may be assumed that the larger variable dominates the behaviour of the function hence,

$$\lim_{|t|\to 0} \chi = L^{\frac{\gamma}{\nu_\parallel}} \mathcal{Q}\left(\frac{1}{|t|^{\nu_\parallel} L}\right) \tag{234}$$

The choice of the factor to take out of the scaling function $\mathcal{P}$ in equation (232) was guided by the knowledge that the correlation length parallel to the field dominates the correlation length perpendicular to it as the transition is approached with field at an angle $\phi = 0°$. The scaling function can be investigated graphically by plotting $\chi L^{-\frac{\gamma}{\nu_\parallel}}$ versus $|t|L^{\frac{1}{\nu_\parallel}}$ for a range of different lattice sizes given that the susceptibility critical exponent is $\gamma = \frac{1}{2}$ and $\nu_\parallel = 1$, equation (226). Figure (56) shows susceptibility data plotted against the theoretical value in the thermodynamic limit with applied field at an angle $\phi = 0°$ as a function of temperature and then the same data replotted using the coordinates identified above so that the data collapses onto the finite size scaling function $\mathcal{Q}$. The previous derivation and figure (56) rely on the choice of $\nu_\parallel$ as the dominant critical length exponent and the precise collapse of the susceptibility data over a range of lattice sizes validates this for applied field at an angle that introduces only commensurate loops into the lattice.

Incommensurate Field Behaviour In the case where the applied field direction creates incommensurate strings the susceptibility shows oscillations similar to the behaviour observed by Bhattacharjee and Nagle in the specific heat when the shape factor $R < \infty$. In that case the behaviour was influenced by the lattice shape changing from an infinitely wide and short lattice with a continuous specific heat to an infinitely tall and thin lattice with a specific heat discretized into delta functions. This change in shape can be considered to have the effect of changing the dominant correlation length so that the scaling hypothesis for susceptibility should be re-derived with $\nu_\perp$ replacing $\nu_\parallel$. If $\nu_\perp$ was dominant in the system then $\chi L^{-\frac{\gamma}{\nu_\perp}} = \chi L^{-1}$ and $|t|L^{\frac{1}{\nu_\perp}} = |t|L^2$ would be the correct variables for the ordinate and abscissa and the scaling behaviour would be expected to cross over from one regime to the other as the applied field angle is reduced from the commensurate angle $\phi = 60°$ where

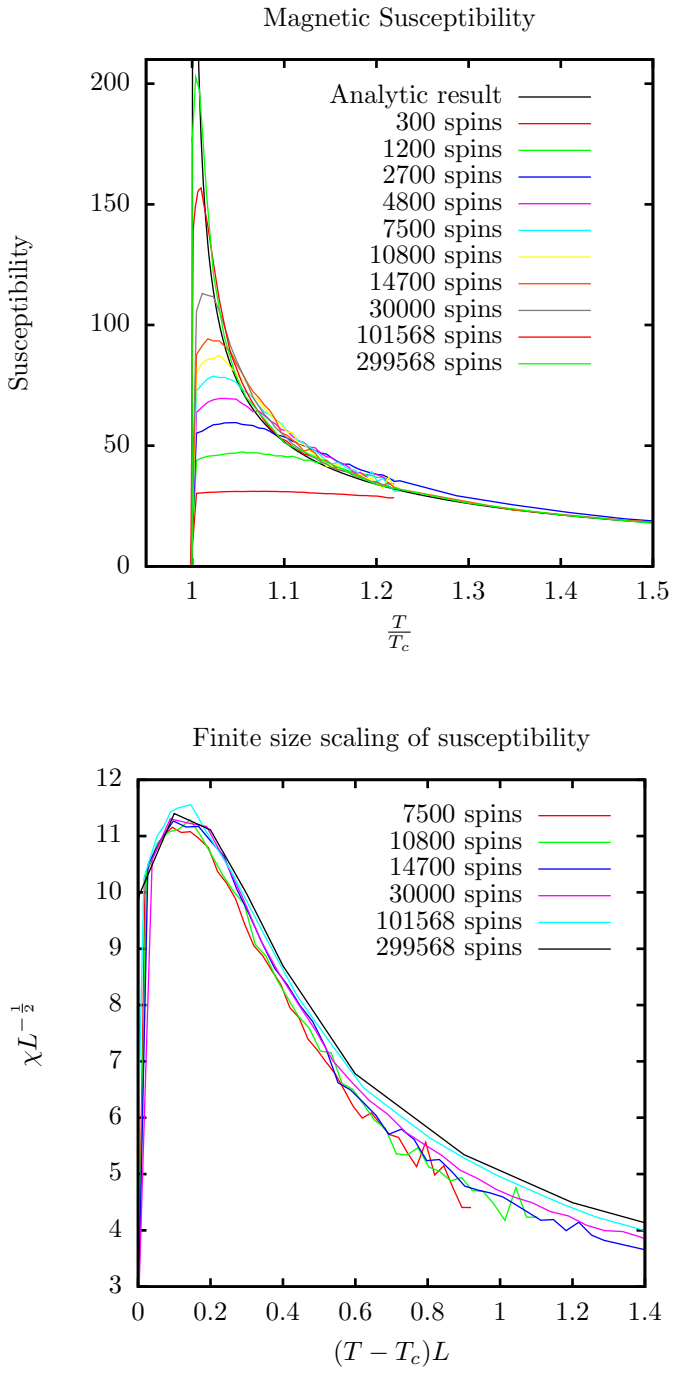

Figure 56: Top: Susceptibility of the kagome ice lattice with field at an angle $\phi = 0°$ for lattice sizes covering three orders of magnitude. Also shown is the theoretical susceptibility in the thermodynamic limit. Bottom: The same data plotted as a function of scaling variables to show the behaviour of the finite size scaling function. The remarkable collapse of the data over a large range of lattice sizes validates the assertion that the correlation length parallel to the field direction dominates the behaviour of the system.

$W = 0$ which is equivalent to $R = \infty$ toward smaller, incommensurate values of ϕ where W increases toward L which, at an angle just above $\phi = 0°$ is equivalent to $R = 0$.

The behaviour of the crossover is continuous as the field angle is reduced from $\phi = 60°$ toward $\phi = 0°$ but as ϕ crosses from positive to negative at exactly $0°$ the system displays a discontinuous change back to $W = 0$. Commensurate loops may be placed into the lattice along the three principal lattice directions and this is expected to occur at the three field angles when the Kasteleyn transition temperature is suppressed to zero. The only exceptions to this occur when the field is parallel to the three angles bisecting the principal directions, $\phi = 0°$, $120°$ and $240°$, so that the finite size scaling behaviour interpolates from the $\Gamma = \infty$ to the $\Gamma = 0$ and back to the $\Gamma = \infty$ limit as the field is rotated through one continuous topological sector except for the angle precisely in the middle of that sector.

Figure (57) shows the results of plotting the susceptibility at field angles of $\phi = 55°$, $45°$, $30°$ and $20°$ as a function of the finite size scaling variables in a similar manner to figure (56) but comparing the behaviour when the parallel and the perpendicular critical correlation length exponent are each chosen to characterise the system. At $\phi = 55°$ the winding number is small and the best collapse of the susceptibility occurs when the system is characterised by a dominant parallel correlation length critical exponent, figure (57) top left, but at $\phi = 20°$ the loops are quite incommensurate, the winding number is becoming large and the best collapse of the susceptibility occurs when the system is characterised by a dominant perpendicular correlation length critical exponent, figure (57) bottom right. At the intermediate values $\phi = 45°$ and $\phi = 30°$ the collapse of the susceptibility is poor using either exponent and the system is strongly in the crossover region between the two dominant regimes.

The quantised nature of the system for $\Gamma \neq \infty$ is a clear signature of a topological phase transition and this behaviour is present for all applied field angles however the gaps between the quantised values decrease as a function of the commensurability and so the system appears continuous at $\Gamma = \infty$. As the field angle is reduced from

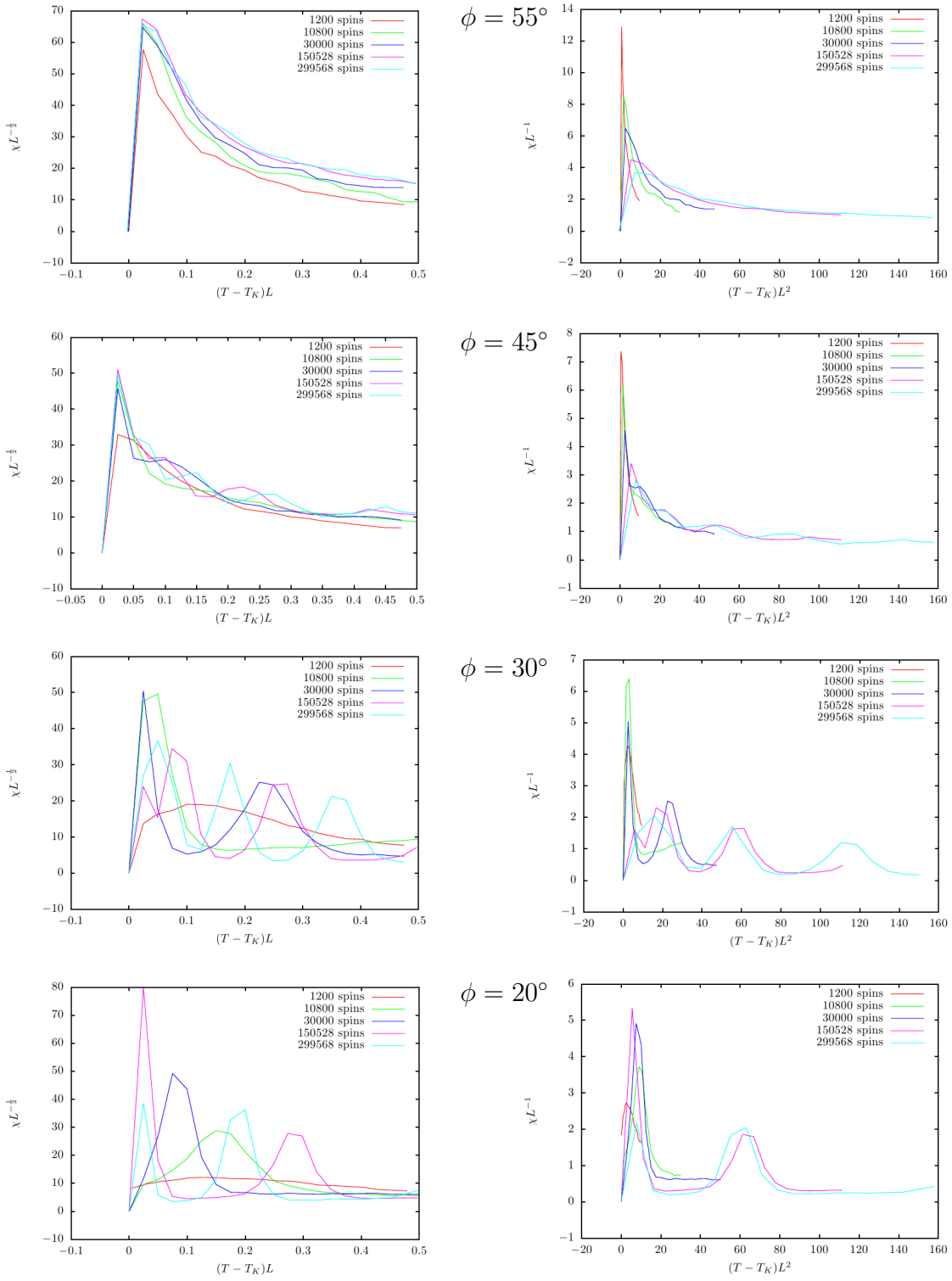

Figure 57: The finite size scaling function for the magnetic susceptibility exhibits a crossover between two limiting behaviours dictated by the critical correlation length exponents. The field is applied at an angle of $\phi = 55°$, $45°$, $30°$ and $20°$ for images from the top to the bottom of the figure. The left column is plotted with scaling variables suitable for the correlation length parallel to the field to be dominant and the right column is plotted with scaling variables suitable for the correlation length perpendicular to the field to be dominant. The behaviour of the susceptibility collapses well in the top left and bottom right figures but with field at intermediate angles neither figure collapses onto a single function.

$\phi = 60°$ toward $\phi = 0°$ the susceptibility scaling function exhibits sharper peaks and tends to zero in the region between these peaks in an exact parallel of the behaviour found by Bhattacharjee and Nagle for lattices with reducing shape factors, see figure (54).

An applied field on the kagome ice plane at an angle $|\phi| < 20°$ but $\phi \neq 0°$ creates a model where, whilst the topological constraint is strictly enforced, many properties are expected to be quantized. I have shown that under these conditions the susceptibility and specific heat display discrete peaks tending to delta functions caused by the discrete addition of topological objects (loops of spins) to the lattice and this will also cause the energy and magnetisation to evolve in corresponding steps and plateaus. This model permits an investigation of a system that exhibits discrete topological steps reminiscent of the quantum Hall effect [119]. The quantisation within the kagome ice model is due to the discrete topological states that the system may occupy just as the quantum Hall effect is due to the occupation of quantised Landau levels however the objects populating the states, topological loop excitations, are somewhat different to the charge excitations or electrons in the quantum Hall effect and it will be interesting to examine how far the analogy between the two can be extended. Further investigation of this effect in kagome ice and its comparison to the quantum Hall effect will be revealing and it is hoped that the discovery of a magnetic system in which to experiment with this phenomenon will be a useful addition and provide a strong motivation to realise an observation of this effect in an experimental system.

7 Square Ice Results

In this chapter a planar model of spin ice that is strictly 2D is discussed in contrast to the kagome ice model examined in the previous chapter. The basis for the model is a square network of interacting spins that has been used to describe the square ice nanoarrays which have become a recent topic of interest in the field of frustrated magnetism [68]. This model was introduced in chapter (1) however a brief recapitulation is now presented. This is followed by a discussion of the computational version of this model and the results that have been obtained using it. Finally a connection is made to an experimental square ice nanoarray that is well described by the particular version of the square ice model presented here.

7.1 A Square Ice Model

When viewed parallel to any of the orthogonal Euclidean axis directions each tetrahedron on the spin ice pyrochlore lattice appears as a square and the four spins on that tetrahedron appear to lie along the body diagonals between the centre and the four corners of the square. Previously in this work each discussion of the spin ice lattice or one of its relatives has focussed on a single tetrahedron (or triangle) as the primary unit and the spins have been placed onto this framework but in this case it is most useful to consider the spins on a tetrahedron (or square) as the primary unit with the lattice framework as secondary. Each group of four spins then defines a vertex where they lie in a right angled cross configuration as illustrated in figure (58)[3].

The spins inherit an Ising nature from spin ice hence there are sixteen spin combinations possible on a vertex and this is a sixteen vertex model. It will be shown later in this chapter that it is convenient to model the spin magnitude as a function of temperature,

$$S = \left(1 - \frac{T}{T_{\mathrm{C}}}\right)^{\beta} \tag{235}$$

where $\beta = \frac{1}{3}$ and $T_{\mathrm{C}} = 230$ K. This function makes the spin magnitude dependent

[3]This figure was inspired by a similar one in reference [32].

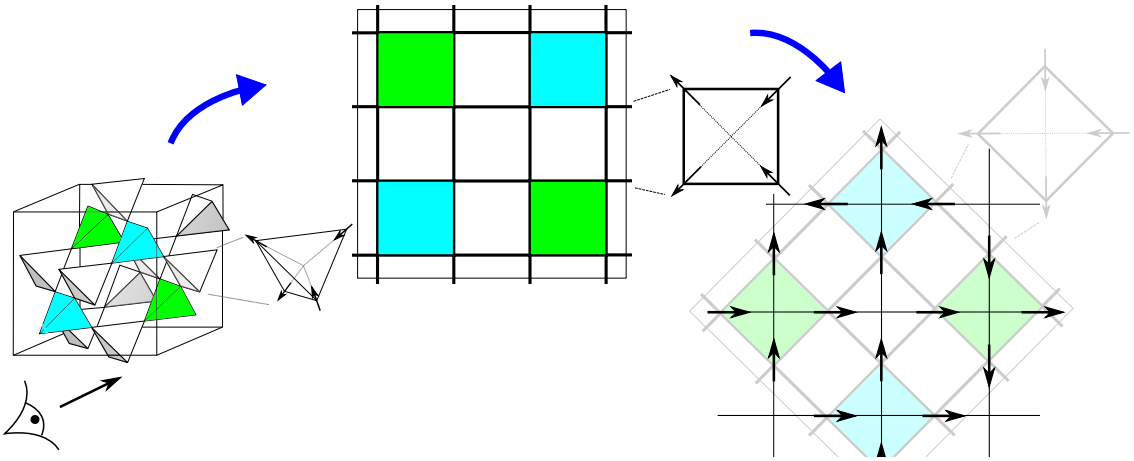

Figure 58: When the spin ice pyrochlore lattice is viewed parallel to one of the cubic directions (left) each tetrahedron appears as a square and the four spins on the tetrahedron form a perpendicular cross (right).

on temperature, reducing it to zero at $T = T_{\mathrm{C}}$.

Topological considerations separate the sixteen vertices into four groups according to the net moment of each vertex as shown in figure (59) where the vertex labelling used there will be continued throughout this chapter.

7.1.1 Spin Interactions

The interaction between the spins is defined to be a ferromagnetic exchange interaction and the Hamiltonian for the model is the same as for the full spin ice model,

$$\mathcal{H}_{ex} = -\kappa_0 \sum_{i,j(n.n.)} \mathbf{S}_i \cdot \mathbf{S}_j - \sum_i \mathbf{H} \cdot \mathbf{S}_i \tag{236}$$

where $\mathbf{H}$ is an externally applied magnetic field and the vector spin $\mathbf{S}_i = \sigma S \mathbf{d}_i$. The pseudo spin variable $\sigma = \pm 1$ dictates the Ising nature of the spins and the direction vectors are $\mathbf{d}_h = (1, 0)$ and $\mathbf{d}_v = (0, 1)$ where odd numbered spins use $\mathbf{d}_v$ and even numbered spins use $\mathbf{d}_h$. The interaction parameter κ_0 is a positive constant to be derived. It will be shown that the important parameter in the model is the ratio $\frac{\kappa_0}{T_{\mathrm{C}}}$ rather than the exchange parameter alone.

Long range interactions are generally a better theoretical approximation to reality than a purely exchange based model. Often a point dipole interaction is employed to model the spin interaction energy however this requires the assumption that the entire dipole moment is concentrated onto a point and the distance between dipoles

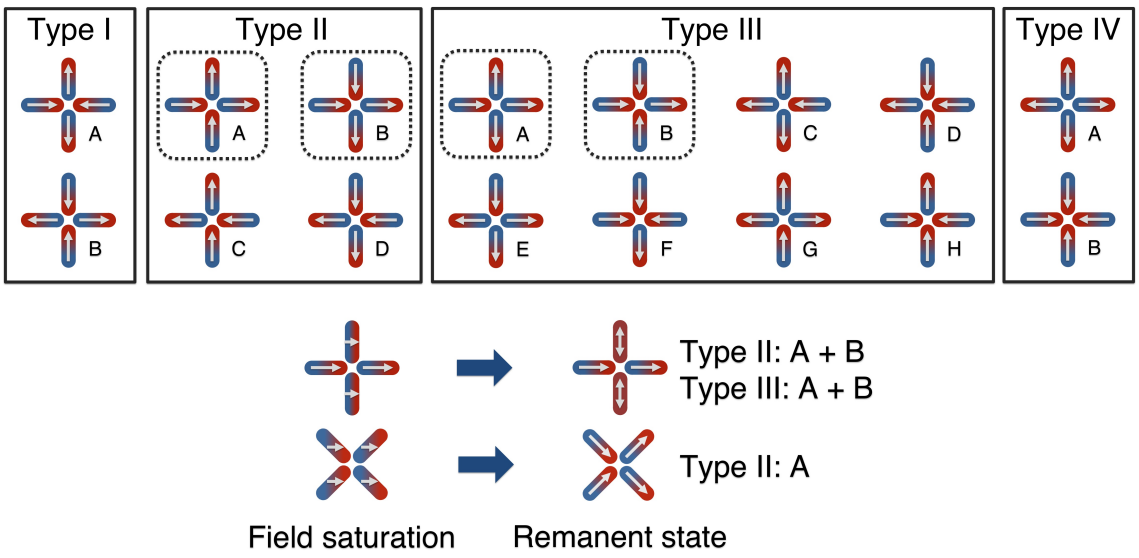

Figure 59: Above: The sixteen vertex configurations of the square ice model grouped into four types by topological considerations of the moment generated by the spins. Below: Experimental application of a saturation field in different directions causes the vertices to relax into specific types at remanence (discussed later in the chapter).

is large compared to their magnitude. This is generally a good approximation for atomic systems and theoretical models but the square ice model constructed here was designed with the idea in mind that it should be relevant to nanoarrays where the pseudospins are large with respect to their separation so that their shape becomes an important consideration.

An alternative long ranged interaction that is more appropriate is a 'dumbbell' model with a positive and negative magnetic charge, or more precisely pole, located at either end of the spin and a coulombic interaction between poles, see figure (60). Retaining the shape of the moment as opposed to condensing it into a point dipole allows this interaction model to be accurate even when the distance between poles is less than the average distance between spins. In general the field produced by a magnetised bar is equivalent to the field produced by a positive and negative pole separated by a distance equal to the length of that bar. This suggests that the dumbbell model will represent the spins well but permit closer investigation of the interactions between them.

The pole interaction energy is given by,

$$E_{coul} = -\frac{\mu_0}{4\pi d} Q_1 Q_2 \tag{237}$$

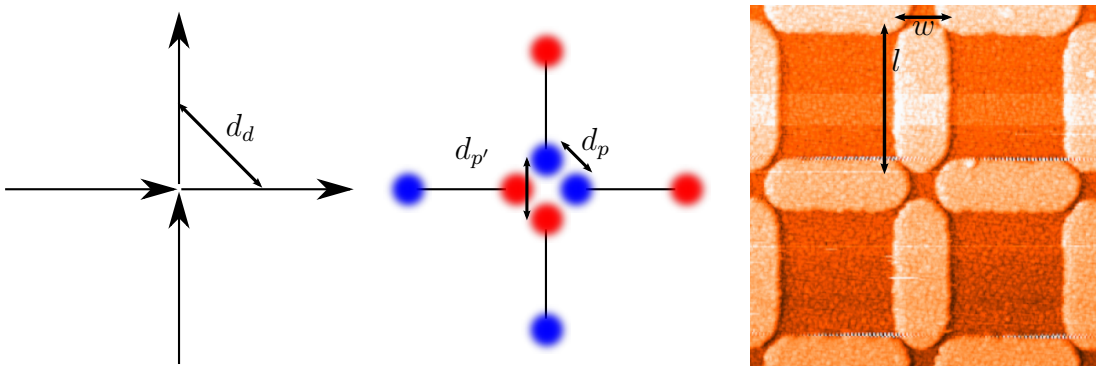

Figure 60: Different representations of a vertex of the square ice model. Left: A vector spin model in the Type IIA configuration. The distance between the centre of two spins, d_d, is required for a point dipole interaction calculation. Centre: Each spin has been replaced by a dumbbell with a positive (red) and negative (blue) charge at either end. These magnetic poles interact in a coulombic manner over the nearest neighbour distance, d_p, and the next nearest neighbour distance, $d_{p'}$, both of which are less than d_d. Right: AFM image of the experimental nanoarray discussed later in the chapter showing a single vertex that could be represented using the spin or dumbbell models. The length of the magnetic island is $l = 750$ nm and the width is $w = 250$ nm.

Considering a single pair of charges estimates a suitable value for the nearest neighbour coupling of the spins which may then be used in the simpler exchange model during simulations. Anticipating the link to the experimental nanoarray, it will be shown that the experimental pole density defines this quantity as,

$$\kappa_0 = 530 \text{ K} \tag{238}$$

The energy of a vertex quickly provides information about the possible lattice ordering hence this is a useful quantity to examine in addition to the interaction between a pair of charges. To calculate the energy of a vertex it is necessary to include the four interactions at the nearest neighbour separation, d_p, and the two interactions at the next nearest neighbour separation, $d_{p'}$ where $d_{p'} = \sqrt{2}d_p$.

Table (4) shows the average energies of the four vertices as a function of the exchange and coulomb interactions where all common factors have been gathered into κ and τ respectively as the purpose of this analysis is not to calculate exact values but to examine the relative energy levels of the vertices. In particular this shows

Vertex type	**Average exchange energy** (κ)	**Average coulomb energy** (τ)
Type I	2	$\sqrt{2}-4$
Type II	-2	$-\sqrt{2}$
Type III	0	0
Type IV	2	$\sqrt{2}+4$

Table 4: Average energies of the square ice vertices as a function of the exchange and coulombic interaction parameters. All other variables in the interactions have been gathered into the variables κ and τ.

that whilst the exchange model selects a four-fold degenerate groundstate formed from the Type II vertices, the coulomb interactions generate a doubly degenerate groundstate formed from the Type I vertices only. Dipolar interactions also select the Type I vertices as most energetically favourable and further these vertices can be arranged in a unique staggered groundstate; however, they are not observed in experimental square ices where the lowest accessible configuration appears to be composed of Type II vertices [68, 71].

In order to reflect this experimental behaviour the energy of the Type I vertices was raised above that of the Type II vertices and for simplicity set equal to the energy of the Type IV vertices to give a single energy scale,

$$E_{III} - E_{II} = E_{IV} - E_{III} = \kappa \tag{239}$$

With the Type II vertices lowest in energy the square ice model created here is equivalent to a model with nearest neighbour exchange interactions only where the exchange parameter is calculated from the coulombic charge interactions as described above. This produces a four-fold degenerate groundstate hence this model is referred to as the four-vertex model.

7.1.2 Ising chain approximation

The square lattice of N' spins is composed of perpendicular chains of spins of length $N = \frac{\sqrt{N'}}{2}$, see figure (61) and, due to the nearest neighbour only interactions which enforce the four-vertex model, this geometry prevents any chain interacting with another because the dot product of perpendicular spins is necessarily equal to zero.

Therefore a simple model for the lattice behaviour is an array of uncoupled 1D chains of Ising spins. Ising's exact solution of the chain model in the thermodynamic limit showed that it does not support a phase transition at any temperature and he discontinued working on the model as he considered it uninteresting [39]. In light of the development of the theory of phase transitions to consider finite size effects and experimental systems such as nanoarrays approaching the limit where finite size effects may dominate the experimental behaviour, the properties of finite size systems have become increasingly relevant and the subject of investigation. The 1D Ising chain displays very strong finite size effects related to its zero temperature quantum critical point that are at the root of the magnetisation behaviour of the square ice nanoarray. Belokon' *et al.* have investigated these effects and provide an expression for the average mean square spin excess (equation (90), reproduced here for convenience) which is a measure of the order present in a system of length N [41].

$$< M^2 > (T) = \frac{1}{N} \frac{1 - \tanh(K^N)}{1 + \tanh(K^N)} \exp(2K) \tag{240}$$

here $K = \frac{J}{T}$ is their exchange interaction parameter scaled by the temperature.

Their work shows that for a chain of finite length at high temperature, $K \to 0$, the magnetisation tends to the limit of the paramagnetic component only but there will always be a temperature below which the spins in the chain will preferentially align parallel to each other and the mean square spin excess will become unity. In the absence of anisotropy a completely aligned chain may still flip between the positive and negative directions along the Ising axis so that the average magnetisation remains zero at all temperatures. By measuring the mean square of the magnetisation it is possible to extract information about the degree of alignment of the spins only without the effects of their direction obscuring the result. A melting temperature, T_{M}, is defined using the magnetisation to label the point at which the ferromagnetic spin order starts to deviate from its maximum value,

$$M(T_{\mathrm{M}}) = 0.99 M(T = 0) \tag{241}$$

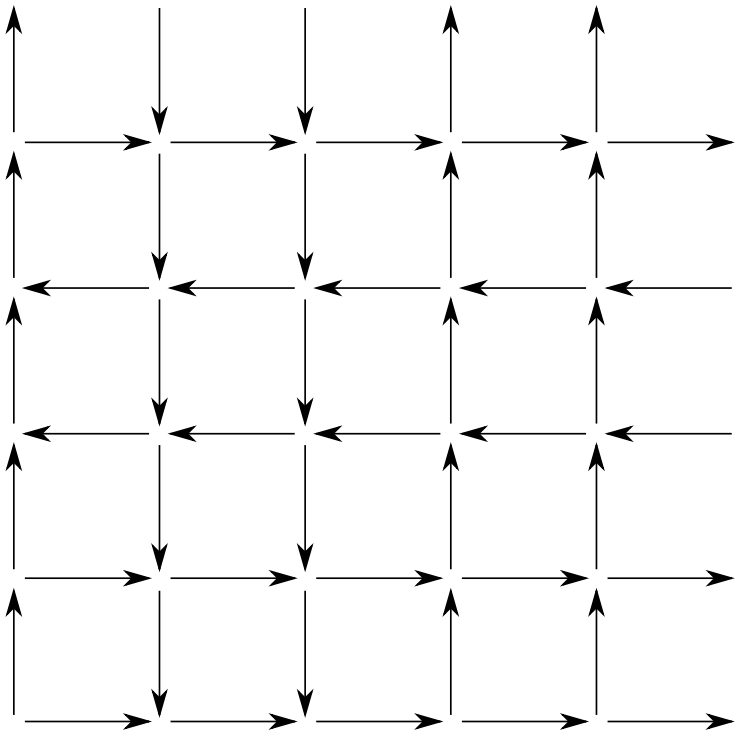

Figure 61: A possible spin configuration for the square lattice where all horizontal and vertical chains of spins are ferromagnetically aligned but the lattice magnetisation is small, $M \approx 0.2M_{max}$. As the lattice size increases this factor is able to be reduced further and for a reasonable system size the lattice magnetisation can quickly approach zero.

however this is not a critical temperature in the classical sense as no thermodynamic phase transition occurs. Belokon' *et al.* find the melting temperature scales logarithmically to zero with the system size so that for all reasonable systems there is a finite temperature region in which the spins are ordered, however in the thermodynamic limit the melting temperature approaches zero in agreement with Ising's original result.

$$T_{\mathrm{M}}^{-1} \propto \ln N \tag{242}$$

We find that similar behaviour to varying the length of the chain can be achieved by varying the strength of the interaction between the spins. In this work the magnitude of the chain magnetisation in the x and y directions is defined as,

$$M_{chx} = \frac{1}{N^2} \sum_{row\, i=1}^{N} \left| \sum_{col\, j=1}^{N} \mathbf{S}_{ij} \right| \tag{243}$$

$$M_{chy} = \frac{1}{N^2} \sum_{col\, i=1}^{N} \left| \sum_{row\, j=1}^{N} \mathbf{S}_{ij} \right| \tag{244}$$

This expression condenses the magnetisation in each horizontal or vertical chain into a single vector and then sums over rows or columns for the x or y chain magnetisation respectively taking the absolute value of the vector to produce a magnitude of the total magnetisation in each direction. The normalised magnetisation of a chain of spins, $\tilde{M}_{sim}$, is defined as the thermal average of the root mean square of the x and y components,

$$\tilde{M}_{sim} = \left\langle \sqrt{\frac{M_{chx}^2 + M_{chy}^2}{2S^2}} \right\rangle \tag{245}$$

and simulations of this quantity for different values of the ratio $\frac{\kappa}{T_{\mathrm{C}}}$ are displayed alongside the analytic expression of Belokon' *et al.* for a single chain (equation(240)) in figure (62). In the work of Belokon' *et al.* unit length spins with a constant interaction are used at all temperatures hence it is necessary to replace their interaction parameter, $K = \frac{J}{T}$, with $\kappa = \frac{\kappa_0 S^2}{T}$ to correspond with the macrospins of reducing length, equation (235), used in the square ice simulation model. The inset of this figure shows the melting temperature, equation (241), as a function of the ratio $\frac{\kappa_0}{T_{\mathrm{C}}}$ calculated analytically using equation (240) with the adapted interaction pa-

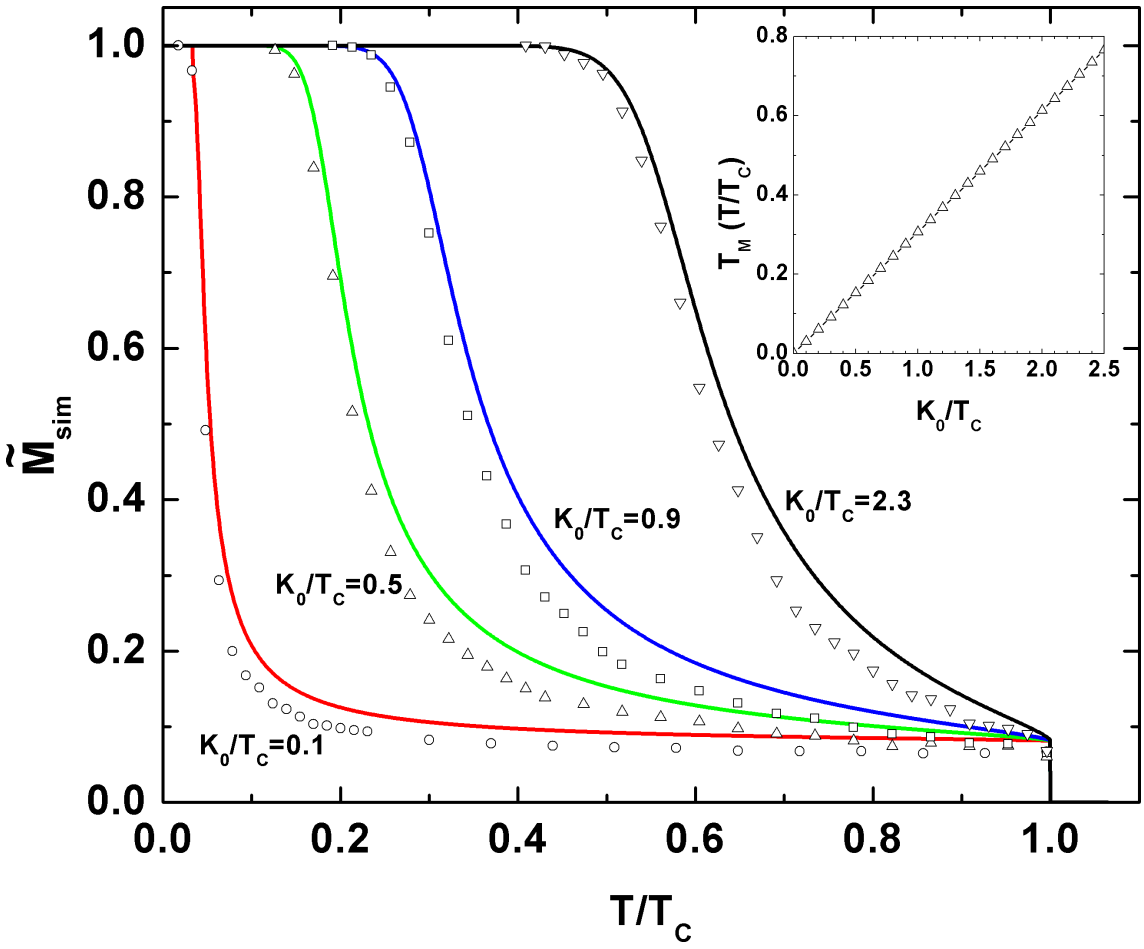

Figure 62: The magnetisation of a chain of spins averaged over all chains in the square ice model. The magnetisation is defined as the root mean square of the simulated chain magnetisation in the x and y directions as described in the text in order to reflect only the ordering of the spins and not their direction. Inset: Defining the melting temperature at the point when the magnetisation is 99% of its maximum, its value can be estimated with respect to a fixed temperature in terms of the spin interaction which shows a linear relationship compared to the logarithmic dependence on chain length.

rameter previously described. The melting of the chain magnetisation occurs at lower temperatures for both increasing chain length and decreasing $\frac{\kappa_0}{T_C}$ however the dependence is different and the melting temperature varies linearly with the ratio rather than logarithmically with chain length as in equation (242). Over the range of interaction strengths shown the relationship is described by,

$$T_M = 0.307\kappa_0 \tag{246}$$

The data presented in figure (62) were simulated on a lattice of $N' = 45000$ spins so that the chains were of length $N = 150$. The simulations were initiated with the

lattice spin configuration randomly generated and all simulations used 1×10^9 MC steps to equilibrate the lattice at each temperature before taking measurements 20000 times with a single MC step between measurements. In order to achieve smooth variation of the magnetisation it was necessary to equilibrate the lattice for this surprisingly large number of steps, however Belokon' *et al.* suggest that in the ordered regime when the exchange interaction between the spins is strong there will be a relaxation time into the ferromagnetic states which scales exponentially with chain length and may therefore be very long [41].

7.2 Comparison of the Model and an Experimental Nanoarray

The extensive research effort dedicated to frustrated materials was given a new avenue for exploration by the publication of Wang *et al.* in 2006 which describes a magnetic nanoarray with statistical properties similar to the spin ice model [68]. Nanoscale fabrication techniques allow experimentalists to construct nanoarrays of magnetic islands which behave as single domain pseudospins and can be geometrically frustrated just as the crystal structure may cause geometric frustration of atomic scale materials. Unlike in these classical frustrated materials, experimental probes can interrogate individual elements of nanoarrays and therefore follow the microscopic evolution of the lattice. The bespoke fabrication of such nanoarrays also allows researchers to include defects at specific locations, accurately vary the lattice geometry or use different construction materials to alter the lattice properties to more precisely suit their purposes.

Making use of the freedom to vary the construction material, Kapaklis *et al.* have overcome one of the problems associated with magnetic nanoarrays [45]. Up until their work the standard construction material was Permalloy with the drawback that its Curie temperature of ~ 900 K prevents any thermal activity in the lattice at reasonable experimental temperatures. As discussed in chapter (2) this means that complicated procedures must be employed to place the lattice into a ground-state configuration and once these have been performed the lattice remains frozen

in that state so there can be no experimental dynamics. Kapaklis *et al.* replace Permalloy with δ-doped Pd(Fe) which has a variable Curie temperature according to the amount of Iron present and choose a composition with a Curie temperature of 230 K. In this way they introduce a thermal energy scale into their experiment at accessible temperatures and accordingly are able to observe dynamic behaviour such as magnetic melting of the lattice.

The square ice model described above was used to model their nanoarray by making the following assumptions. Each island i in the lattice is considered to be composed of n individual unit length microspins, $\mathbf{s}_{i'}$, which are coupled to their neighbours via exchange interactions with a coupling constant J related to T_{C} so that for all temperatures less than the Curie temperature of the material the island may be considered as a single macrospin defined to be of unit length,

$$\mathbf{S}_i = \frac{\sum_{i' \in i} \mathbf{s}_{i'}}{|\sum_{i' \in i} \mathbf{s}_{i'}|} \tag{247}$$

The interaction between the macrospins is long ranged, but the characteristic energy scale may be defined through the interaction between neighbouring spins where the most significant interaction is that between second nearest neighbours which couples spins along a row or column of the lattice. The interaction parameter $\kappa = \kappa_0 m^2$, where κ_0 is of order n^2 and $m^2 = \left\langle \left(\frac{1}{n}\sum_{i' \in i} \mathbf{s}_{i'}\right)^2 \right\rangle$ is a thermally averaged order parameter for a single island so that the Hamiltonian remains,

$$\mathcal{H}_{ex} = -\kappa_0 \sum_{i,j(n.n.)} \mathbf{S}_i \cdot \mathbf{S}_j - \sum_i \mathbf{H} \cdot \mathbf{S}_i \tag{248}$$

and written in this way the macrospins have a reducing magnitude as in the initial model (equation (235) reproduced here) as experimentally the macrospins are expected to be reduced by fluctuations or the possible formation of sub-domains as temperature is increased.

$$S = \left(1 - \frac{T}{T_{\mathrm{C}}}\right)^{\beta} \tag{249}$$

The values $T_{\mathrm{C}} = 230$ K and $\beta = \frac{1}{3}$ are chosen to match the behaviour of the unpatterned film from which the nanoarray was created, see figure (63).

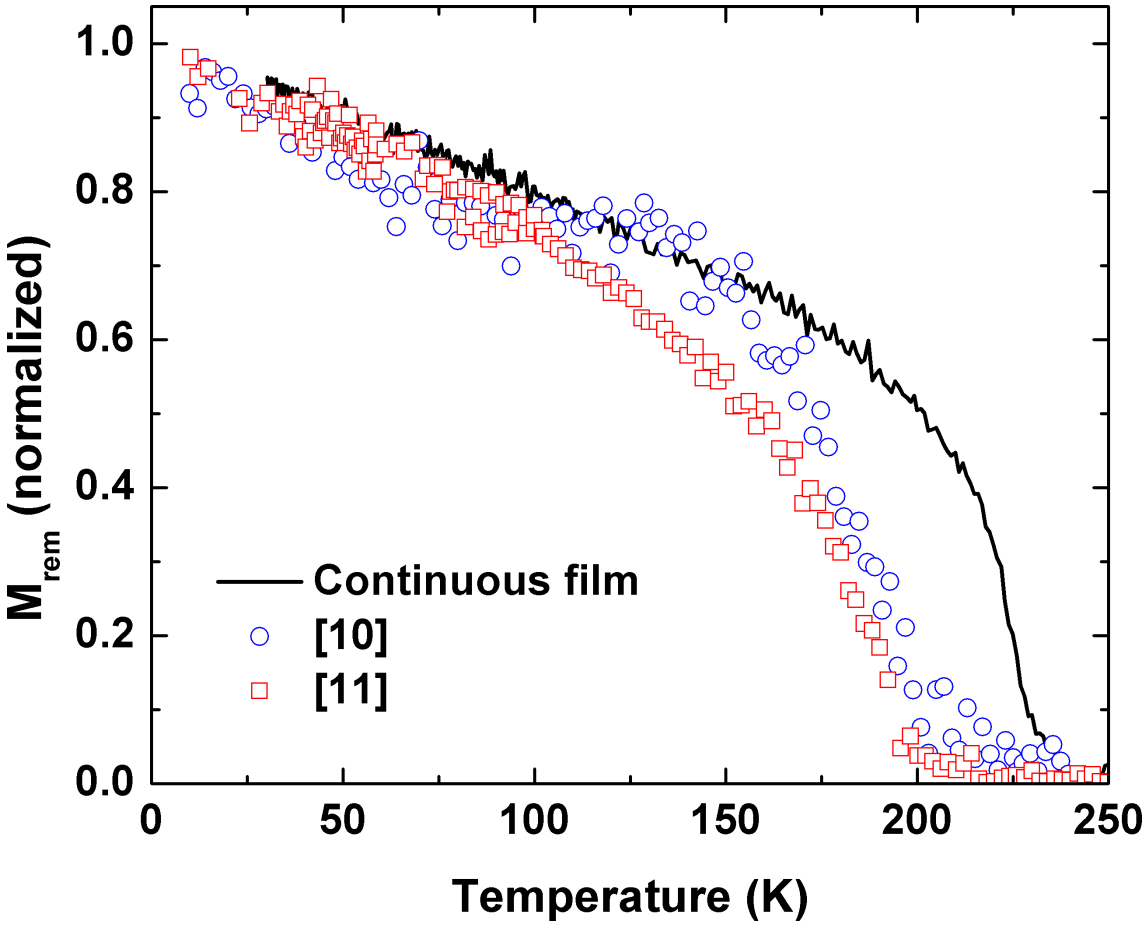

Figure 63: Experimental remanent magnetisation measured by Kapaklis *et al.* [45] of the unpatterned film used to create the nanoarray compared with that of the patterned film after applying a magnetic field along the $[1, 0]$ and $[1, 1]$ directions. The collapse of the magnetisation in the patterned films with field in both directions at a lower temperature than the unpatterned film indicates that inter-island interactions rather than inter-spin interactions are controlling the behaviour of the lattice and creating thermal dynamics.

The temperature range of the simulations is restricted to below the Curie temperature hence the microspins do not need to be individually modelled, leaving a square lattice of Ising macrospins. The macrospins are modelled with Ising characteristics to reflect the magnetic shape anisotropy which minimises the island energy when the moment lies parallel to the long axis of the island which is certainly a good approximation over most of the experimental temperature range; however, it may not be strictly upheld near to the Curie temperature of the material as discussed below.

Introducing the thermally averaged quantity m^2 implies that the thermal fluctuations of the macro and micro degrees of freedom are decoupled, which ensures that the two sets can order independently. This should be an excellent approximation for the islands like those of this nanoarray which consist of $\sim 10^7$ microspins. The ordering of the macrospins must therefore occur at a temperature $T_{\mathrm{M}} \leq T_{\mathrm{C}}$, as even in the limit $\kappa_0 \gg J$, the term in m^2 ensures that $\kappa < J$ at T_{C}. Experimental observations show a wide range of temperature, $T_{\mathrm{M}} < T < T_{\mathrm{C}}$, in which the ordered islands are decoupled but remain well described as single macrospins.

Returning to the calculation of spin interactions using a dumbbell model, equation (237), SQUID measurements provide values for the pole density to give,

$$\frac{E_{coul}}{k_B} = \pm 532 \text{ K} \tag{250}$$

hence the value $\kappa_0 = 530$ K is used in the model as an approximation of the strength of the exchange interaction between neighbouring spins. Similarly, theoretical analysis suggested the Type II vertices were the degenerate groundstate using exchange interactions but long range interactions, such as those relevant experimentally, favour the Type I vertices to form a unique groundstate. Previous experiments on square ice nanoarrays have not observed disordered vertices relaxing into these states which was explained by the lack of a thermal energy scale preventing the lattice from accessing them due to dynamical reasons [120]; however, this unique groundstate is still not observed in the dynamic square ice nanoarray of Kapaklis *et al.* indicating that its absence is not of dynamical origin.

Due to the large moment of each island ($\sim 10^7 \mu_B$) we suggest that even in experimental zero field conditions a magnetic field as small as the Earth's magnetic field or a stray laboratory field will be likely to decrease the energy of the Type II vertices below that of the Type I vertices at all temperatures leaving the Type II vertices as the effective degenerate groundstate manifold. It is also possible that other effects disfavour the Type I vertices such as multipolar interactions which are known to destabilise Type I vertices in arrays of compass needles [121, 122] or other departures from ideality such as non-Ising spin behaviour. In order to capture this behaviour the model retains exchange interactions only which has the effect of selecting a four-fold degenerate groundstate, just as a vertex model with long ranged interactions would if the Type I vertices are artificially raised in energy.

This model was used to simulate the effect of applying a saturation field in the $[1,0]$ or $[1,1]$ directions and then allowing the lattice to relax to a remanent magnetic state, see figure (59) for the possible saturated and remanent vertex configurations. With field applied parallel to the $[1,1]$ direction there is only one remanent state observed, Type II: A of figure (59) whilst when the field is applied parallel to the $[1,0]$ direction it is parallel to two of the four islands at a vertex and causes them to order in that direction but it is perpendicular to the other two and does not affect them. At low temperatures the uncoupled spins are still energetically biased to take a co-parallel arrangement, hence, of the four vertices that satisfy the requirement of the coupled spins, only those of Type II will be selected at remanence but as the temperature is increased the lattice will be increasingly likely to relax into Type III vertices as well. Experimentally this procedure was quantified by measuring the ratio of the remanent to saturation values of the magnetisation. By considering their ratio the individual island order parameter is removed from the calculation and the results reflect only the ordering behaviour of the macrospins. Geometric arguments indicate that for single domain spins in the low temperature groundstate

configurations specified above these ratios will be

$$\frac{M_{rem}}{M_{sat}}\left([1,1]\right) = \frac{1}{\sqrt{2}} \tag{251}$$

$$\frac{M_{rem}}{M_{sat}}\left([1,0]\right) = \frac{1}{2} \tag{252}$$

Figure (64) shows the experimental results of measuring this ratio compared to the simulated behaviour using the four-vertex model. The experimental measurements were obtained by cycling the field up to a saturation value of approximately 50 mT in the appropriate direction and then allowing the system to return to remanence before taking the measurement, whilst the simulation results were produced by equilibrating a square lattice from a random configuration in a small field ($H = 0.03\kappa_0$) in order to reproduce the effect of the lattice magnetisation due to saturation.

At low temperatures the experiment and simulation correspond very well indicating that the four-vertex model is appropriate to describe the nanoarray and reciprocally that the nanoarray is well described using single domain macrospins. The ratios then remain at their maximum values until temperatures where thermal fluctuations have reduced the value of the macrospin order parameter, m^2, to less than half of its initial value. At approximately 125 K the value of the ratio for the $[1,1]$ case starts to decrease and at approximately 150 K the ratio for the $[1,0]$ case also appears to start decreasing. Both ratios fall close to zero before the Curie temperature of the material is reached indicating that the magnetisation of the lattice is governed by the macrospin interactions rather than the microspin interactions of the material from which it is constructed.

The behaviour of the simulation deviates from the experimental results at temperatures close to the Curie temperature where it is possible that an Ising spin model does not capture the full detail of the nanoarray as when the thermal energy scale is close to the interaction energy scale behaviour such as transverse fluctuations of the spins may emerge. This is also the region in which the moment of the islands is rapidly decreasing and the possibility of domain wall formation or other effects that reduce the effective spin moment cannot be ruled out.

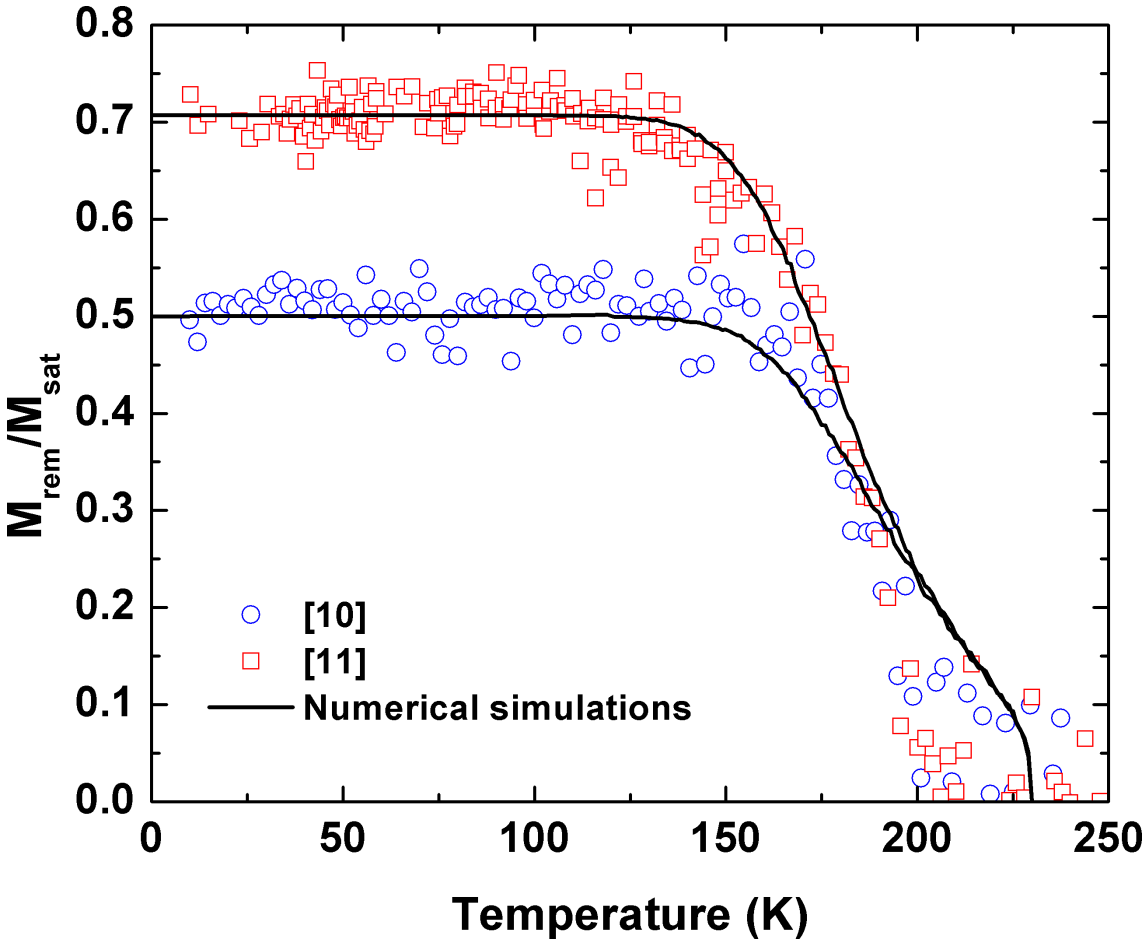

Figure 64: The ratio of remanent to saturation magnetisation after a field has been applied parallel to the $[1,0]$ and $[1,1]$ directions compared with the simulation results using the four-vertex model. Geometric considerations indicate that the low temperature values ought to be 0.5 and 0.707 respectively as observed. Further detail of the measurements is provided in the text.

Both experiment and simulation show that the ratio for the $[1,1]$ field appears to decrease at a lower temperature than for the $[1,0]$ field. The value starts to decrease at temperatures when spin flips excite the low temperature ordered Type II vertices to Type III vertices. In the case of the $[1,0]$ ordered vertices there are transitions to Type III vertices that preserve the magnetisation component along the field direction and so would not reduce the ratio, for example Type II: A or B becoming Type III: A or B, however any excitation of the $[1,1]$ ordered state will necessarily flip a spin against the field direction reducing the value of the remanent magnetisation as illustrated in figure (59). Therefore although the temperature at which the thermal energy scale is large enough to induce spin flip excitations in the lattice is the same regardless of the direction of the saturation field and both ratios start to decrease at the same temperature, the ratio for the $[1,1]$ case appears to start at a lower temperature because it decreases faster than the ratio for the $[1,0]$ case.

In order to explain the melting behaviour of the lattice it is useful to recall the observation that to a first approximation the perpendicular islands do not interact and hence the lattice can be represented as an uncoupled array of Ising chains. The theoretical analysis of finite size Ising chains, summarised by figure (62), indicates that there will always be a temperature, T_{M}, at which each chain will order and so given a saturation field to coordinate the ordering direction this will produce a finite, maximum magnetisation at low temperature which 'melts' at a temperature dependent on the ratio of the interaction parameter and the critical temperature of the lattice material.

With regard to the inset of figure (62), the interaction parameter for the model, equation (238), predicts a melting temperature of,

$$T_M = 0.307 \times 530 = 163 \text{ K} \tag{253}$$

which is in close agreement with the experimental observations illustrated in figure (64). This lends further support to the interpretation of the experimental square ice nanoarray as an ensemble of uncoupled Ising spin chains with strong finite size

effects.

The results of this simulation provide a convincing argument for the finite size ordering phenomenon in Ising chains as the source of the melting behaviour observed in the square ice nanoarray. In the experimental protocol the magnetisation is measured via hysteresis loops so that at every temperature the system is ordered by the field and then allowed to relax to a remanent state where the magnetisation value is recorded. Figure (59) shows that when the field is applied parallel to the $[1, 1]$ direction there is a unique ordering of all chains whereas with field applied parallel to the $[1, 0]$ direction half of the chains are parallel to the field and are fully ordered but the other half are perpendicular to the field and therefore uncoupled to it. At low temperatures these uncoupled chains will preferentially internally align as any antiparallel spins would indicate the presence of an unfavourable Type III vertex, but at higher temperatures the state of the uncoupled chains is indeterminate as the ordering is unlikely to be strictly enforced. The analysis of the finite size Ising model shows that by varying the interaction strength between the spins it is always possible to enforce a melting temperature below any finite fixed temperature for a chain of spins and so it is also possible to create a melting temperature for the square ice lattice below the Curie temperature of the material where the macrospin description remains valid and the islands are well defined single domains but the chain ordering is no longer present. At this point the magnitude of the magnetisation of each chain falls to a value reflecting only its paramagnetic component and the magnetisation of the lattice falls to approximately zero as this paramagnetic component fluctuates in direction within each chain and averages to zero over the entire lattice.

The ability to separate the macro and microspin behaviour in this lattice due to the variable interaction strength between islands is novel and, at the time of writing, unique. We have shown using a simple model that this permits the realisation of an artificial square spin ice array which demonstrates true thermal dynamics for the first time and should allow future investigations to pursue properties which have previously been inaccessible in nanoarrays.

8 Review, Conclusions and Future Perspectives

The process of writing a program to simulate the kagome ice model was begun by simulating some simpler models such as the Ising square lattice, see figure (4), which provided an opportunity to learn the elementary aspects of Monte Carlo simulation, how to use a linux operating system and to discover many important and basic procedures such as how to create a makefile to compile a series of code files, how to operate a computer remotely using ssh and how to run simulations on a computer cluster through a resource management interface. A significant amount of theoretical work was also conducted during these early stages of the doctorate which became the basis of chapter (3) and chapter (4) describing Monte Carlo simulation, magnetic neutron scattering and the Kasteleyn transition.

The primary objective of creating a kagome ice model was to include a loop update algorithm into the simulation as this is necessary to accurately model kagome ice in the region close to the Kasteleyn transition, see chapter (5). Implementation of the loop algorithm was based on previous work in this area [35, 112] but proceeded very slowly due to the complex nature of the problem. Whilst attempting to construct the algorithm the probabilities required at each vertex of the lattice were initially calculated utilising a single triangle as the minimum unit within the lattice however it was discovered that this leads to a non-Markovian algorithm and a pair of triangles together is the appropriate choice. At this stage an investigation into using both forward and backward moves during the construction of each loop was conducted and whilst this is always necessary below the Kasteleyn transition temperature it was not found to be required above it. Other factors considered were whether it was necessary to move only one defect or both and whether to use the short or long loop condition [107] to close the loops. Eventually, with the applied field in a specific, high symmetry direction ($[\bar{1}, \bar{1}, 2]$) only as this provided a large simplification in their calculation, the probabilities necessary for the construction of the loop were derived and the program began to produce meaningful results.

In order to link the simulation results back to experiment a separate program was constructed with the purpose of creating simulated neutron scattering maps. This

process also involved some detours such as changing the entire scattering calculation from a coordinate frame local to the kagome plane to laboratory coordinates describing the complete spin ice lattice and back again in an attempt to isolate errors in the program, after resolving these problems it was finally possible to simulate results that were comparable to experimental data.

The work of Moessner and Sondhi [33] provided a starting point for much of the project on kagome ice but whilst it contains much new (at that time) and important information it predicts contrasting neutron scattering maps to those produced in this work. The first conclusion that was drawn using the simulations presented in chapter (6) was that there was a mistake in Moessner and Sondhi's analytically calculated neutron scattering maps, see section (6.1), due to a sign error. Excepting this difference in intensity in some regions of the scattering plane the simulated results corresponded well to their data and also with other experiments and simulations providing reassurance that any further results had a firm basis.

To complete the kagome ice simulation model it was necessary to fully implement the loop algorithm so that it would work for arbitrary field angle. Whilst writing this part of the algorithm it was not clear whether it was necessary for the periodic boundary conditions to twist with the applied field in order to always provide a 'shortest path' across the lattice or to remain fixed at a half twist so that the lattice shape is a rhombohedron but the connectivity is that of an square. The solution was found to be a fixed twist which is somewhat counter-intuitive as the theory of the transition indicates that at this temperature a single string should enter the lattice, and discontinuous magnetisation results at the transition which had been ascribed to the loop winding around the lattice twice because of simple periodic boundary conditions had prompted the introduction of twisted periodic boundary conditions, although as explained in section (6.2.3) the requirement for the first loop to cross the lattice once is in fact only true at the boundaries between the three continuous topological sectors and at their bisectors and in all other cases the loop will wind around the lattice more than once.

Returning again to the objective of linking simulation results back to experi-

ment, one of the first things achieved with the fully functional kagome ice model and neutron scattering program was a simulation of a region of the phase space corresponding to one of the recent experiments carried out on a kagome ice material. The comparison of these data sets, figure (51), has begun the process of understanding the Kasteleyn phase transition in kagome ice materials in earnest and is leading toward one of the original goals of this work which was to provide a simulation model that can act as a 'roadmap' for future experimental work highlighting areas of interest and supplementing in cases where experimental challenges make obtaining data difficult. So far it has not been possible to observe the Kasteleyn transition experimentally but the kagome ice model has gone some way toward explaining unusual experimental results such as twisted scattering peaks, section (6.1.4), and has made predictions regarding unexpected scattering patterns that it may be possible to observe, figure (49).

A separate area of work in this doctorate has been established in the field of magnetic nanoarrays. In order to support some exciting results measured on an artificial square ice nanoarray I have written a program which contains a square lattice of Ising spins and produces thermodynamic data as a function of temperature for different applied fields. This simplified model permits an identification of the experimental square ice nanoarray as an ensemble of finite size 1D Ising spin chains from which it can be shown that for materials where the ratio of the microscopic interaction strength and the macroscopic island interaction strength can be controlled it is always possible to create a region in which the behaviour of the islands is decoupled from their constituent material and reflects only the lattice geometry. This work is presented in chapter (7).

Recent simulation results of the thermodynamic properties and particularly the critical finite size scaling behaviour of the kagome ice model at the Kasteleyn transition, see section (6.2), have shown some interesting features. This work required the loop algorithm to be complete as it is focussed on the angular dependence of kagome ice with respect to an applied field and is only present when the topological constraint is strictly enforced. I have shown that kagome ice has the property of

being exceptionally sensitive to the applied field direction, such that misalignment of tenths of a degree is significant, and that under certain conditions it displays a biaxial magnetisation, section (6.2.1).

I have scrutinised the work of Bhattacharjee and Nagle who investigated the finite size scaling behaviour of a model of hardcore dimers on a brick lattice in section (6.2.2). Using the topological quantum number identifying the kagome ice state, I have derived a measure in equation (220) which I refer to as the commensurability of a topological excitation that enables a straightforward comparison between their work and the kagome ice model. Investigation of the temperature dependent finite size scaling function as the angle of the applied field on the kagome plane is altered shows a dramatic variation in response and I propose that under suitable conditions some measurable properties of this model such as the magnetisation will exhibit a series of topological steps and plateaus reminiscent of the quantum Hall effect, see section (6.2.4). The extent of the analogy between these two effects is currently unclear but this intriguing result certainly calls for further investigation of this area, in both a theoretical and experimental setting.

The sensitivity of kagome ice with respect to applied field and their proximity to the Kasteleyn transition suggest that experimental observation of many of the results presented in this work will be challenging. The appeal of these objectives is that whilst spin ice has been extensively experimentally explored these phenomena retain some interesting questions in a well understood model and, as with many results in spin ice, these may prove illuminating to other areas of research. The Kasteleyn transition and more generally the characteristics of topological materials remain relatively obscure but as properties that occur due to simple constraints they may be expected to become more widely relevant as they are better understood.

The importance of the topological nature of spin ice has grown with the recent discovery of magnetic monopoles in both the spin ice model and spin ice materials and it may be that the quantisation shown by some properties of kagome ice under appropriate conditions serves to increase this further. Kagome ice is now in a position to act as a rich testing ground for the effects of topological constraints and

topological phase transitions.

The outcome of intriguing theoretical predictions that occur in experimentally challenging conditions has become a theme of this work; however, I hope that this serves to spur future investigators forward rather than deter them.

References

[1] H. B. Callen. *Thermodynamics and an Introduction to Thermostatistics*. Wiley, 2^{nd} edition, 1985. 1, 1.3.3, 1.3.4

[2] C. Kittel. *Introduction to Solid State Physics*. J. Wiley and Sons, Inc., 7^{th} edition, 1996. 1.1, 1.1, 1.3.3, 3.2

[3] N. Goldenfeld. *Lectures on Phase Transitions and the Renormalisation Group*. Addison-Wesley, 1^{st} edition, 1995. 1.3.2, 1.4.1

[4] H. E. Stanley. *Introduction to Phase Transitions and Critical Phenomena*. Oxford University Press, 1^{st} edition, 1971. 1.1

[5] A. I. M. Rae. *Quantum Mechanics*. Institute of Physics Publishing, 4^{th} edition, 2002. 1.1

[6] P. A. M. Dirac. The Quantum Theory of the Electron. *Proceedings of the Royal Society A*, 117:610–624, 1928. 1.1

[7] B. Cowen. *Topics in Statistical Mechanics*. Imperial College Press, 1^{st} edition, 2005. 1.3.2

[8] P. Weiss. L'Hypothèse du champ moléculaire et la propriété ferromagnétique. *Journal de Physique*, 6:661–690, 1907. 1.3.3

[9] P. Weiss. *Zeitschrift für Physik*, 9:358, 1908. 1.3.3

[10] L. D. Landau. *Physik zür Sowjetunion*, 11:26, 1937. 1.3.3

[11] L. D. Landau and E. M. Lifschitz. *Statistical Physics, Part 1*. Butterworth Heinemann, Oxford, 3^{rd} edition, 1980. 1.3.3

[12] E. A. Guggenheim. *Journal of Chemical Physics*, 13:253, 1945. 1.3.4

[13] S. T. Bramwell, P. C. W. Holdsworth, and J.-F. Pinton. Universality of rare fluctuations in turbulance and critical phenomena. *Nature*, 396:552–554, 1998. 1.3.4

[14] M. J. Harris, S. T. Bramwell, D. F. McMorrow, T. Zeiske, and K. W. Godfrey. Geometrical Frustration in the Ferromagnetic Pyrochlore $Ho_2Ti_2O_7$. *Physical Review Letters*, 79(13):2554–2557, 1997. 1.4, 1.4.2, 1.4.2, 1.4.2, 1.4.3, 2.1, 2.2.1

[15] L. Onsager. Crystal Statistics. 1. A Two-dimensional Model with an Order-Disorder Transition. *Physical Review*, 65:117–149, 1944. 1.4.1

[16] C. N. Yang. The Spontaneous Magnetisation of a Two-Dimensional Ising Model. *Physical Review*, 85(5):808–816, 1952. 1.4.1, 4

[17] S. Naya. On the Spontaneous Magnetisations of Honeycomb and Kagomé Ising Lattices. *Progress of Theoretical Physics*, 11(1):53–62, 1954. 4, 1.4.1

[18] I. Syôzi. Statistics of Kagomé Lattice. *Progress of Theoretical Physics*, 6(3):306–308, 1951. 4, 1.4.1

[19] A. Taroni. *Theoretical Investigations of Two-Dimensional Magnets*. PhD thesis, University College London, 2007. 1.4.1

[20] K. Husimi and I. Syôzi. The Statistics of Honeycomb and Triangular Lattice. I. *Progress of Theoretical Physics*, 5(2):177–186, 1950. 1.4.1

[21] I. Syôzi. The Statistics of Honeycomb and Triangular Lattice. II. *Progress of Theoretical Physics*, 5(3):341–351, 1950. 1.4.1

[22] A. Codello. Exact Curie temperature for the Ising model on Archimedean and Laves lattices. *Journal of Physics A: Mathematical and Theoretical*, 43:385002, 2010. 1.4.1

[23] G. H. Wannier. Antiferromagnetism. The Triangular Ising Net. *Physical Review*, 79(2):357–364, 1950. 1.4.1

[24] K. Kanô and S. Naya. Antiferromagnetism. The Kagomé Ising Net. *Progress of Theoretical Physics*, 10(2):158–172, 1953. 1.4.1

[25] A. S. Wills, R. Ballou, and C. Lacroix. Model of localized highly frustrated ferromagnetism: The *kagomé* spin ice. *Physical Review B*, 66:144407, 2002. 1.4.1, 1.4.1, 1.4.3, 4.2

[26] P. W. Anderson. Ordering and Antiferromagnetism in Ferrites. *Physical Review*, 102(54):1008–1014, 1956. 1.4.2, 1.4.2

[27] G. Toulouse. Theory of frustration effect in spin-glasses. 1. *Communications on Physics*, 2(4):115–119, 1977. 1.4.2

[28] S. T. Bramwell and M. J. Harris. Frustration in Ising-type spin models on the pyrochlore lattice. *Journal of Physics: Condensed Matter*, 10:L215–L220, 1998. 1.4.2

[29] R. Siddharthan, B. S. Shastry, and A. P. Ramirez. Spin ordering and partial ordering in holmium titanate and related systems. *Physical Review B*, 63:184412, 2001. 1.4.2

[30] B. C. den Hertog and M. J. P. Gingras. Dipolar Interactions and Origin of Spin Ice in Ising Pyrochlore Magnets. *Physical Review Letters*, 84(15):3430–3433, 2000. 1.4.2

[31] R. G. Melko, B. C. den Hertog, and M. J. P. Gingras. Long-Range Order at Low Temperatures in Dipolar Spin Ice. *Physical Review Letters*, 87(6):067203, 2001. 2.1

[32] S. T. Bramwell and M. J. P. Gingras. Spin Ice State in Frustrated Magnetic Pyrochlore Materials. *Science*, 294:1495–1501, 2001. 13, 2.1, 3

[33] R. Moessner and S. L. Sondhi. Theory of the [111] magnetisation plateau in spin ice. *Physical Review B*, 68:064411, 2003. 1.4.2, 1.5, 4.1, 4.4, 4.5, 6.1, 40, 6.1.1, 6.1.1, 41, 6.1.3, 6.1.3, 8, A, A

[34] R. Siddharthan, B. S. Shastry, A. P. Ramirez, A. Hayashi, R. J. Cava, and S. Rosenkranz. Ising Pyrochlore Magnets: Low-Temperature Properties, "Ice Rules," and Beyond. *Physical Review Letters*, 83(9):1854–1857, 1999. 1.4.2

[35] R. G. Melko and M. J. P. Gingras. Monte Carlo studies of the dipolar spin ice model. *Journal of Physics: Condensed Matter*, 16:R1277–R1319, 2004. 1.4.2, 2.1, 5, 6.1.5, 8

[36] J. D. Bernal and R. H. Fowler. A Theory of Water and Ionic Solution, with Particular Reference to Hydrogen and Hydroxyl Ions. *Journal of Chemical Physics*, 1(8):515–548, 1933. 1.4.2, 2.1

[37] M. J. Harris, S. T. Bramwell, P. C. W. Holdsworth, and J. D. M. Champion. Liquid-Gas Critical Behaviour in a Frustrated Pyrochlore Ferromagnet. *Physical Review Letters*, 81(20):4496–4499, 1998. 1.4.2

[38] A. J. Macdonald, P. C. W. Holdsworth, and R. G. Melko. Classical topological order in kagome ice. *Journal of Physics: Condensed Matter*, 23:164208–164217, 2011. 1.4.3, 2.3, 2.3, 6.2.3

[39] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31, 1925. 1.4.5, 7.1.2

[40] R. J. Baxter. *Exactly Solved Models in Statistical Mechanics*. Academic Press, 1982. 1.4.5

[41] V. I. Belokon', K. V. Nefedev, O. A. Goroshko, and O. I. Tkach. Superparamagnetism in the 1D Ising Model. *Bulletin of the Russian Academy of Sciences: Physics*, 74(10):1413–1416, 2010. 1.4.5, 7.1.2, 7.1.2

[42] P. W. Kasteleyn. Dimer Statistics and Phase Transitions. *Journal of Mathematical Physics*, 4(2):287–293, 1963. 1.5, 4.1

[43] K. Matsuhira and Z. Hiroi and T. Tayama and S. Takagi and T. Sakakibara. A new macroscopically degenerate ground state in the spin ice compound $Dy_2Ti_2O_7$ under a magnetic field. *Journal of Physics: Condensed Matter*, 14:L559–L565, 2002. 1.5, 2.2

[44] T. Fennell, S. T. Bramwell, D. F. Mcmorrow, P. Manual, and A. R. Wildes. Pinch points and Kasteleyn transitions in kagome ice. *Nature Physics*, 3:566–572, 2007. 1.5, 2.2.1, 17, 6.1, 40, 6.1.1

[45] V. Kapaklis, U. B. Arnalds, A. Harman-Clarke, E. T. Papaioannou, M. Karimipour, P. Korelis, A. Taroni, P. C. W. Holdsworth, S. T. Bramwell, and

B. Hjörvarsson. Melting artificial square spin-ice. Submitted for publication, 2011. 1.5, 2.3, 7.2, 63

[46] W. F. Giauque and J. W. Stout. The Entropy of Water and the Third Law of Thermodynamics. The Heat Capacity of Ice from 15 to 273°K. *Journal of the American Chemical Society*, 58(7):1144–1150, 1936. 2.1

[47] L. Pauling. The Structure and Entropy of Ice and of Other Crystals with Some Randomness of Atomic Arrangement. *Journal of the American Chemical Society*, 57:2860–2864, 1935. 2.1

[48] E. O. Wollan, W. L. Davidson, and C. G. Shull. Neutron Diffraction Study of the Structure of Ice. *Physical Review*, 75(9):1348–1352, 1949. 2.1

[49] J. C. Li, V. M. Nield, D. K. Ross, R. W. Whitworth, C. C. Wilson, and D. A. Keen. Diffuse neutron-scattering study of deuterated ice Ih. *Philosophical Magazine B*, 69(6):1173–1181, 1994. 2.1

[50] H. W. J. Blöte, R. F. Wielinga, and W. J. Huiskamp. Heat-Capacity Measurements On Rare-Earth Double Oxides $R_2M_2O_7$. *Physica*, 47:549–568, 1969. 2.1

[51] A. P. Ramirez, A. Hayashi, R. J. Cava, R. Siddharthan, and B. S. Shastry. Zero-point entropy in 'spin ice'. *Nature*, 399:333–335, 1999. 14, 2.1

[52] K. Matsuhira, Y. Hinatsu, and T. Sakakibara. Novel dynamical magnetic properties in the spin ice compound $Dy_2Ti_2O_7$. *Journal of Physics: Condensed Matter*, 13:L737–L746, 2001. 2.1

[53] J. Snyder, B. G. Ueland, J. S. Slusky, H. Karunadasa, R. J. Cava, A. Mizel, and P. Schiffer. Quantum-Classical Reentrant Relaxation Crossover in $Dy_2Ti_2O_7$ Spin Ice. *Physical Review Letters*, 91(10):107201, 2003.

[54] M. Orendáč, J. Hanko, E. Čižmár, and A. Orendáčová. Magnetocaloric study of spin relaxation in dipolar spin ice $Dy_2Ti_2O_7$. *Physical Review B*, 75:104425, 2007. 2.1

[55] C. Castelnovo, R. Moessner, and S. L. Sondhi. Magnetic monopoles in spin ice. *Nature*, 451:42–45, 2008. 2.1, 2.3, 4.3, 5.5

[56] L. D. C. Jaubert and P. C. W. Holdsworth. Signature of magnetic monopole and Dirac string dynamics in spin ice. *Nature Physics*, 5(4):258–261, 2009. 2.1, 4.3

[57] T. Fennell, P. P. Deen, A. R. Wildes, K. Schmalzl, D. Prabhakaran, A. T. Boothroyd, R. J. Aldus, D.F. McMorrow, and S. T. Bramwell. Magnetic Coulomb Phase in the Spin Ice $Ho_2Ti_2O_7$. *Science*, 326(5951):415–417, 2009. 2.1, 2.2.1, 17, 3.2.1

[58] D. J. P. Morris, D. A. Tennant, S. A. Grigera, B. Klemke, C. Castelnovo, R. Moessner, C. Czternasty, M. Meissner, K. C. Rule, J.-U. Hoffman, K. Kiefer, S. Gerischer, D. Slobinsky, and R. S. Perry. Dirac Strings and Magnetic Monopoles in the Spin Ice $Dy_2Ti_2O_7$. *Science*, 326:411–414, 2009. 2.1, 2.2.1, 17, 6.1.5

[59] A. P. Ramirez. Strongly Geometrically Frustrated Magnets. *Annual Review of Materials Science*, 24:453–480, 1994. 2.2

[60] X. Obradors, A. Labarta, A. Isalgué, J. Tejada, J. Rodriguez, and M. Pernet. Magnetic frustration and lattice dimensionality in $SrCr_8Ga_4O_{19}$. *Solid State Communications*, 65(3):189 – 192, 1988. 2.2

[61] A. S. Wills and A. Harrison. Structure and magnetism of hydronium jarosite, a model Kagomé antiferromagnet. *Journal of the Chemical Society - Faraday Transactions*, 92:2161–2166, 1996. 2.2

[62] R. H. Colman, A. Sinclair, and A. S. Wills. Comparisons between Haydeeite, $\alpha - Cu_3Mg(OD)_6Cl_2$, and Kapellasite, $\alpha - Cu_3Zn(OD)_6Cl_2$, Isostructural $S = \frac{1}{2}$ Kagome Magnets. *Chemistry of Materials*, 22:5774–5779, 2010. 2.2

[63] Z. Hiroi, K. Matsuhira, S. Takagi, and T. Tayama T. Sakakibara. Specific Heat of Kagomé Ice in the Pyrochlore Oxide $Dy_2Ti_2O_7$. *Journal of the Physical Society of Japan*, 72(2):411–418, 2003. 2.2, 15

[64] S. T. Bramwell, M. J. Harris, B. C. den Hertog, M. J. P. Gingras, J. S. Gardner, D. F. McMorrow, A. R. Wildes, A. L. Cornelius, J. D. M. Champion, R. G. Melko, and T. Fennell. Spin Correlations in $Ho_2Ti_2O_7$: A Dipolar Spin Ice System. *Physical Review Letters*, 87(4):047205, 2001. 2.2.1, 16

[65] Y. Tabata, H. Kadowaki, K. Matsuhira, Z. Hiroi, N. Aso, E. Ressouche, and B. Fåk. Spin correlation in kagomé ice state: Neutron scattering study of the dipolar spin ice $Dy_2Ti_2O_7$ under magnetic field along $[1, 1, 1]$. *Journal of Magnetism and Magnetic Materials*, 310:1311–1313, 2007. 2.2.1, 18

[66] H. Kadowaki, N. Doi, Y. Aoki, Y. Tabata, T. J. Sato, J. W. Lynn, K. Matsuhira, and Z. Hiroi. Observation of Magnetic Monopoles in Spin Ice. *Journal of the Physical Society of Japan*, 78(10):103706, 2009. 2.2.1

[67] T. Fennell. Kagome Ice. 2010. 2.2.1, 19, 50

[68] R. F. Wang, C. Nisoli, R. S. Freitas, J. Li, W. McConville, B. J. Cooley, M. S. Lund, N. Samarth, C. Leighton, V. H. Crespi, and P. Schiffer. Artificial 'spin ice' in a geometrically frustrated lattice of nanoscale ferromagnetic islands. *Nature*, 439:303–306, 2006. 2.3, 20, 7, 7.1.1, 7.2

[69] M. Tanaka, E. Saitoh, H. Miyajima, T Yamaoka, and Y. Iye. Magnetic interactions in a ferromagnetic honeycomb nanoscale network. *Physical Review B*, 73:052411, 2006. 2.3

[70] J. Cumings. Artificial ice goes thermal. *Nature Physics*, 7:7–8, 2011. 20

[71] E. Mengotti, L. J. Heyderman, A. Fraile Rodriguez, A. Bisig, L. Le Guyader, F. Nolting, and H-B. Braun. Building blocks of an artifical kagome spin ice: Photoemission electron microscopy of arrays of ferromagnetic islands. *Physical Review B*, 78:144402, 2008. 2.3, 7.1.1

[72] J. P. Morgan, A. Stein, S. Langridge, and C. H. Marrows. Thermal ground-state ordering and elementary excitations in artifical magnetic square ice. *Nature Physics*, 7:75–79, 2011. 2.3

[73] S. Ladak, D. E. Read, G. K. Perkins, L. F. Cohen, and W. R. F. Branford. Direct observation of magnetic monopole defects in an artificial spin ice system. *Nature Physics*, 6:359–363, 2010. 2.3

[74] L. A. Mól, R. L. Silva, R. C. Silva, A. R. Pereira, W. A. Moura-Melo, and B. V. Costa. Magnetic monopole and string excitations in two-dimensional spin ice. *Journal of Applied Physics*, 106:063913, 2009.

[75] E. Mengotti, L. J. Heyderman, A. F. Rodriguez, F. Nolting, R. V. Hügli, and H-B. Braun. Real-space observation of emergent magnetic monopoles and associated Dirac strings in artificial kagome spin ice. *Nature Physics*, 7:68–74, 2010. 2.3

[76] J. Kerr. On Rotation of the Plane of Polarisation by Reflection from the Pole of a Magnet. *Philosophical Magazine*, 3(19):321–343, 1877. 2.3.1

[77] J. Kerr. Reflection of Polarised Light from the Equatorial Surface of a Magnet. *Philosophical Magazine*, 5(30):161–177, 1878. 2.3.1

[78] P. Weinberger. John Kerr and his effects found in 1877 and 1878. *Philosophical Magazine Letters*, 88(12):897–907, 2008. 2.3.1

[79] H. J. Williams, F. G. Foster, and E. A. Wood. Observation of Magnetic Domains by the Kerr Effect. *Physical Review*, 82(1):119–120, 1951. 2.3.1

[80] N. Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, 15:125–130, 1987. 3

[81] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. 3, 3.1

[82] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, 2^{nd} edition, 2002. 3

[83] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. N. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953. 3.1

[84] V. I. Manousiouthakis and M. W. Deem. Strict detailed balance is unnecessary in Monte Carlo simulation. *Journal of Chemical Physics*, 110:2753–2756, 1999. 3.1, 5.5

[85] J. von Neumann. Various techniques used in connection with random digits. In G. E. Forsyth A. S. Householder and H. H. Germond, editors, *Monte Carlo Method*, pages 36–38. Washington, D.C.: U.S. Government Printing Office, 1951. 3.1

[86] R. Pynn. Neutron Scattering: A Primer. *Los Alamos Science*, 19, 1990. 23

[87] S. T. Bramwell. Neutron Scattering and Highly Frustrated Magnetism. In C. Lacroix, P. Mendels, and F. Mila, editors, *Introduction to Frustrated Magnetism*, pages 45–77. Springer Series in Solid State Sciences, 2011. 3.2, 3.2.1

[88] J. W. Gibbs. *Elements of vector analysis, arranged for the use of students in physics*. New Haven, 1881. 3.2

[89] W. L. Bragg. The Diffraction of Short Electromagnetic Waves by a Crystal. *Proceedings of the Cambridge Philosophical Society*, 17:43–57, 1914. 3.2

[90] R. W. Youngblood, J. D. Axe, and B. M. McCoy. Correlations in ice-rule ferroelectrics. *Physical Review B*, 21(11):5212–5220, 1980. 3.2.1

[91] R. W. Youngblood and J. D. Axe. Polarization fluctuations in ferroelectric models. *Physical Review B*, 23(1):232–238, 1982. 3.2.1

[92] M. E. Fisher. Statistical Mechanics of Dimers on a Plane Lattice. *Physical Review*, 124(6):1664–1672, 1961. 4.1

[93] P. W. Kasteleyn. The Statistics of Dimers on a Lattice. I. The Number of Dimer Arrangements on a Quadratic Lattice. *Physica*, 27:1209–1225, 1961. 4.1

[94] D. A. Huse, W. Krauth, R. Moessner, and S. L. Sondhi. Coulomb and Liquid Dimer Models in Three Dimensions. *Physical Review Letters*, 91(16):167004, 2003. 4.1

[95] J. F. Nagle. Lipid Bilayer Phase Transition: Density Measurements and Theory. *Proceedings of the National Academy of Sciences of the U. S. A.*, 70(12):3443–3444, 1973. 4.1

[96] F. Y. Wu. Remarks on the Modified Potassium Dihydrogen Phosphate Model of a Ferroelectric. *Physical Review*, 168(2):539–543, 1968. 4.1, 4.5

[97] G. R. Allen. Dimer models for the antiferroelectric transition in copper formate tetrahydrate. *The Journal of Chemical Physics*, 60(8):3209–3309, 1974. 4.1

[98] J. F. Nagle. Statistical mechanics of the melting transition in lattice models of polymers. *Proceedings of the Royal Society of London A*, 337:569–589, 1974. 4.1

[99] M. E. Fisher. Walks, Walls, Wetting and Melting. *Journal of Statistical Physics*, 34(5/6):667–729, 1984. 4.1

[100] L. D. C. Jaubert, J. T. Chalker, P. C. W. Holdsworth, and R. Moessner. Three-Dimensional Kasteleyn Transition: Spin Ice in a [100] Field. *Physical Review Letters*, 100:067207, 2008. 4.1, 4.5

[101] L. D. C. Jaubert, J. T. Chalker, P. C. W. Holdsworth, and R. Moessner. Spin Ice under Pressure: Symmetry Enhancement and Infinite Order Multicriticality. *Physical Review Letters*, 105:087201, 2010. 4.2

[102] J. Villain. Insulating Spin Glasses. *Zeitschrift für Physik B*, 33:31–42, 1979. 4.5

[103] J. F. Nagle, C. S. O. Yokoi, and S. M Bhattacharjee. Dimer Models on Anisotropic Lattices. In C. Domb and J. L. Lebowitz, editors, *Phase Transitions and Critical Phenomena*, pages 235–297. Academic Press, 1989. 4.5, 6.2.2

[104] L. D. C. Jaubert, J. T. Chalker, P. C. W. Holdsworth, and R. Moessner. The Kasteleyn transition in three dimensions: spin ice in a [100] field. *Journal of Physics: Conference Series*, 145:012024, 2009. 4.5

[105] R. H. Swendsen and J.-S. Wang. Nonuniversal Critical Dynamics in Monte Carlo Simulations. *Physical Review Letters*, 58(2):86–88, 1987. 5

[106] U. Wolff. Collective Monte Carlo Updating for Spin Systems. *Physical Review Letters*, 62(4):361–364, 1989. 5

[107] G. T. Barkema and M. E. J. Newman. Monte Carlo simulation of ice models. *Physical Review E*, 57(1):1155–1166, 1998. 5, 8

[108] A. Rahman and F. H. Stillinger. Proton Distribution in Ice and the Kirkwood Correlation Factor. *The Journal of Chemical Physics,*, 57(9):4009–4017, 1972. 5

[109] S. V. Isakov, K. Raman, R. Moessner, and S. L. Sondhi. Magnetization curve of spin ice in a [111] magnetic field. *Physical Review B*, 70:104418, 2004. 5

[110] T. C. Germann and K. Kadau. Trillion-atom molecular dynamics becomes a reality. *International Journal of Modern Physics C*, 19(9):1315–1319, 2008. 5.2

[111] This mapping was (re)discovered and suggested by L. D. C. Jaubert. 5.3

[112] L. D. C. Jaubert. *Topological Constraints and Defects in Spin Ice*. PhD thesis, Ecole Normale Supérieure de Lyon, 2009. 5.4, 8

[113] M. E. Zhitomirsky. Octupolar Ordering of classical kagome antiferromagnets in two and three dimensions. *Physical Review B*, 78:094423, 2008. 6.1, 40, 6.1.1

[114] D. A. Garanin and B. Canals. Classical spin liquid: Exact solution for the infinite-component antiferromagnetic model on the *kagomé* lattice. *Physical Review B*, 59(1):443–456, 1999. 6.1.1

[115] S. M. Bhattacharjee and J. F. Nagle. Finite-Size effect for the critical point of an anisotropic dimer model of domain walls. *Physical Review A*, 31(5):3199–3213, 1981. 6.2.2, 54, 6.2.3, 6.2.4

[116] C. S. O. Yokoi, J. F. Nagle, and S. R. Salinas. Dimer Pair Correlations on the Brick Lattice. *Journal of Statistical Physics*, 44(5/6):729–747, 1986. 6.2.4

[117] S. M. Bhattacharjee and J. J. Rajasekaran. Absence of anomalous dimension in vertex models: Semidilute solution of directed polymers. *Physical Review A*, 44(10):6202–6212, 1991. 6.2.4

[118] M. E. Fisher. Interface Wandering in Adsorbed and Bulk Phases, Pure and Impure. *Journal of the Chemical Society: Faraday Transactions 2*, 82:1569–1603, 1986. 6.2.4

[119] T. Ando, Y. Matsumoto, and Y. Uemura. Theory of Hall Effect in a Two-Dimensional Electron System. *Journal of the Physical Society of Japan*, 39:279–288, 1975. 6.2.4

[120] A. Remhof, A. Schumann, A. Westphalen, H. Zabel, N. Mikuszeit, E. Y. Vedmedenko, T. Last, and U. Kunze. Magnetostatic interactions on a square lattice. *Physical Review B*, 137:134409, 2008. 7.2

[121] E. Olive and P. Molho. Thermodynamic study of a lattice of compass needles in dipolar interaction. *Physical Review B*, 58(14):9238–9247, 1998. 7.2

[122] E. Y. Vedmedenko, N. Mikuszeit, H. P. Oepen, and R. Wiesendanger. Multipolar Ordering and Magnetisation Reversal in Two-Dimensional Nanomagnet Arrays. *Physical Review Letters*, 95:207202, 2005. 7.2

Appendices

A Units for Neutron Scattering

The units used by experimentalists and theoreticians sometimes vary when discussing the reciprocal space in which neutron scattering patterns are measured. In this appendix the conventions used in this work are specified along with their relation to those used in reference [33] for clarity.

Written in terms of Miller indices the reciprocal plane of interest is that perpendicular to the [111] direction within the spin ice pyrochlore lattice so that $\hat{x}$ is parallel to $[\bar{1}, 1, 0]$ and $\hat{y}$ is parallel to $[\bar{1}, \bar{1}, 2]$. The sixteen site unit cell of the pyrochlore lattice is defined to have a side of length a so that the vector $[\bar{1}, \bar{1}, 2]$ has a length,

$$k_y = \frac{2\pi}{a}\sqrt{6} \tag{254}$$

Experimentally the first pinch point along the $[\bar{1}, \bar{1}, 2]$ direction moving away from zero is at,

$$k_0 = \frac{4\pi}{a}\sqrt{\frac{2}{3}} = \frac{2}{3}k_y \tag{255}$$

which is the scale used in the simulated neutron scattering maps calculated in this work so that the the first pinch point appears at a value of $\frac{2}{3}$ on the y-axis.

Moessner and Sondhi define a unit length, a', for their analytic scattering maps (see figure (40) and reference [33]) relative to the unit cell of the kagome lattice which is triangular with a side twice the length of the neighbouring spin distance. Within the pyrochlore framework nearest neighbours are separated by, for example, $\frac{a}{4}[1, 1, 0]$ therefore,

$$a' = \frac{a}{\sqrt{2}} \tag{256}$$

The scattering maps they present are stated to be for $|\mathbf{Q}| = \pm 4\pi$ which gives a characteristic scale for their figure of,

$$k = \frac{4\pi}{a}\sqrt{2} \tag{257}$$

This can be verified by considering the ratio of the pinch point position to their

characteristic length,

$$\frac{k_0}{k} = \frac{\frac{4\pi}{a}\sqrt{\frac{2}{3}}}{\frac{4\pi}{a}\sqrt{2}} = \frac{1}{\sqrt{3}} \tag{258}$$

They do not provide axes or a scale on their images however the first pinch point is approximately $\frac{1}{\sqrt{3}}$ above the origin.

B Kagome Ice Code

```
      subroutine avlat

      use defs

      implicit none
          !outputs the average spin direction at each site in the lat
  tice after taking all the snapshots for the neutron scattering.

      integer::s_av_counter1,s_av_counter2,s_av_counter3
      character(3)::s_av_label!,s_av_format*8
      real(kind=dp),dimension(2)::s_av_r
      real(kind=dp),dimension(2),parameter::s_av_a=(/2.0_dp,0._dp/),s
  _av_b=(/-1.0_dp,-r3/)

      !add up the spin vectors at each site
      do s_av_counter1=0,side
         do s_av_counter2=0,side
            do s_av_counter3=1,3
               avlattice(s_av_counter1,s_av_counter2,s_av_counter3)=a
  vlattice(s_av_counter1,s_av_counter2,s_av_counter3)+&
               lattice(s_av_counter1,s_av_counter2,s_av_counter3)
            end do
         end do
      end do

      !if this is the last call then output the lattice
      if(called==snapshots)then
         avlattice=avlattice/snapshots
         counter8=counter8+1
         if(counter8<10)then
            write(s_av_label,'(I1,I1,I1)')0,0,counter8
         elseif(counter8>9.and.counter8<100)then
            write(s_av_label,'(I1,I2)')0,counter8
         else
            write(s_av_label,'(I3)')counter8
         end if
         open(21, file='avlattice'//s_av_label//'.lattice', status='replace')
         do s_av_counter1=0,side
            do s_av_counter2=0,side
               s_av_r=s_av_counter2*s_av_a+s_av_counter1*s_av_b
               if(avlattice(s_av_counter1,s_av_counter2,1)>0)then
                  write(21,'(4E20.3)')s_av_r(1),s_av_r(2)+0.4,0.0,-1.0
               else
                  write(21,'(4E20.3)')s_av_r(1),s_av_r(2)-0.4,0.0,1.0
               end if
               s_av_r=s_av_r+(0.5*s_av_b)
               if(avlattice(s_av_counter1,s_av_counter2,2)>0)then
                  write(21,'(4E20.3)')s_av_r(1)-0.433,s_av_r(2)-0.25,0.
  866,0.5
               else
                  write(21,'(4E20.3)')s_av_r(1)+0.433,s_av_r(2)+0.25,-0
  .866,-0.5
               end if
               s_av_r=s_av_r+(0.5*s_av_a)
               if(avlattice(s_av_counter1,s_av_counter2,3)>0)then
                  write(21,'(4E20.3)')s_av_r(1)+0.433,s_av_r(2)-0.25,-0
  .866,0.5
               else
                  write(21,'(4E20.3)')s_av_r(1)-0.433,s_av_r(2)+0.25,0.
  866,-0.5
               end if
            end do
         end do
         avlattice=0
      end if
```

```
59          close(21)
60
61          end subroutine
```

```
      subroutine bweights

      use defs

      implicit none
      !this routine calculates the boltzmann weights and the probabilities for the loop algorithm so that
          !they can be stored as a lookup table rather than worked out each time. this is for efficiency
          !further detail of the weights, energies etc are in the note from PCWH and the thesis, chap 5

      !calculate the boltzmann weights of the states at this temperature or field
          !the energy is the sum of the spins dotted with field/temperature.
          !only the spins that are in the up triangle are counted. w=exp(beta(h.s))
      wO=exp(dot_product((-S1+S2+S3),field)/temperature)!the long range ordered state
      wL=exp(dot_product((S1+S2-S3),field)/temperature)!from the perspective of the spin at the head of the loop, the loop moving left
      wR=exp(dot_product((S1-S2+S3),field)/temperature)!... or moving right.

      !calculate the two energies
      eps0=2.*r2*field(2)
      eps1=2.*r2*field(1)/r3

      !calculate the probabilities
      pL=exp(eps1/temperature)/(exp(eps1/temperature)+exp(-eps1/temperature)) !prob to go left
      pR=exp(-eps1/temperature)/(exp(eps1/temperature)+exp(-eps1/temperature))  !prob to go right
      pO=pR*wO/wR
          !prob to go back to ordered vertex
      pLR=1.-pO
          !prob to go from L to R (or R to L) vertex
      pLL=1
          !prob to go from L to L (or R to R) vertex

!the probabilities for the different moves in the loop algorithm for interest, numbers refer to the five vertex types
!       p12=pR
!       p13=pR
!       p14=pL
!       p15=pL
!       p21=pO
!       p24=pLR
!       p23=pLL
!       p31=pO
!       p35=pLR
!       p32=pLL
!       p41=pO
!       p42=pLR
!       p45=pLL
!       p51=pO
!       p54=pLR
!       p53=pLL

      end subroutine
```

```
      subroutine check

      use defs

      implicit none
          !this subroutine checks whether the loop can close at this point and possibly flips the tail back for a short loop

!       integer::s_che_counter1,s_che_x1,s_che_y1,s_che_n1,s_che_flstop,s_che_adj

!       fliptail=0

!       if(longloop==0)then

      !******************** need to update this part!!! currently short close is not supported.

          !check if the triangle of the current spin already has the loop going through it
!          if(loop_hist((3*(side+1)*loop_x)+(3*loop_y)+loop_n)==1)then!the loop has been through this spin before, short close with overlap
!             fliptail=1
! !             write(*,*)'closed a short loop with overlap'
!          elseif((loop_hist((3*(side+1)*loop_x)+(3*loop_y)+1))+(loop_hist((3*(side+1)*loop_x)+(3*loop_y)&
!             +2+mod(loop_n-1,2)))==2)then!both the other spins on this triangle are in the loop already, short close
!             fliptail=1
! !             write(*,*)'closed a short loop without overlap'
!          elseif((loop_hist((3*(side+1)*loop_x)+(3*loop_y)+1))+(loop_hist((3*(side+1)*loop_x)+(3*loop_y)&
!             +2+mod(loop_n-1,2)))==1)then!one of the other spins is in the loop...
!             if(loop_y==loop_ylist(1))then
!                if(loop_x==loop_xlist(1))then!...same triangle as the first spin so closed a long loop
!                   loopstop=1
!                   flooplength=loopspins
! !                   write(*,*)'closed a long loop without flipslave, longloop=0'
!                end if
!             end if
!          end if
!       elseif(longloop==1)then
          if(loop_x==cla_x)then!cla_x etc are the coordinates of the neighbouring spins to the first spin in the loop. they are recorded in loopmc in the initialise section
             if(loop_y==cla_y)then
                if(loop_n==cla_n)then!same down triangle as the first spin so closed a long loop
                   loopstop=1
                   flooplength=loopspins
                end if
             end if
          end if
          if(loop_x==clb_x)then
             if(loop_y==clb_y)then
                if(loop_n==clb_n)then!same down triangle as the first spin so closed a long loop
                   loopstop=1
                   flooplength=loopspins
                end if
             end if
```

```
          end if
          if(loop_x==loop_xlist(1))then
             if(loop_y==loop_ylist(1))then
                if(loop_n==loop_nlist(1))then!got back to the first sp
in, must have closed a long loop by retracing the first bit of it
                   loopstop=1
                   flooplength=loopspins
                end if
             end if
          end if
!       end if

!       !if a short loop has closed then flip the tail back to its or
iginal state
!       if(fliptail==1.or.flipall==1)then
! !          write(*,*)'Flipping a tail back'
!          loopstop=1
!          s_che_counter1=1
!          s_che_flstop=0
!          do while(s_che_flstop==0)
!             s_che_x1=loop_xlist(s_che_counter1)
!             s_che_y1=loop_ylist(s_che_counter1)
!             s_che_n1=loop_nlist(s_che_counter1)
!             if(flipall==0)then
!                if(s_che_x1==loop_x)then
!                   if(s_che_y1==loop_y)then!you're on the up triangl
e where the loop closes
!                      if(s_che_n1==loop_n)then!head of loop overlapp
ed tail when closing don't need to flip last spin
!                         s_che_flstop=1
!                         s_che_adj=0
! !                         write(*,*)'overlap, didnt flip last spin'
!                      else
!                         lattice(s_che_x1,s_che_y1,s_che_n1)=-lattic
e(s_che_x1,s_che_y1,s_che_n1)
! !                         write(*,*)'last time, flipped spin',s_che
_x1,s_che_y1,s_che_n1,'back'
!                         s_che_flstop=1
!                         s_che_adj=1
!                      end if
!                   end if
!                end if
!             else
!                if(s_che_counter1>loopspins)then
!                   s_che_flstop=1
!                   s_che_adj=1
!                end if
!             end if
!             if(s_che_flstop/=1)then
!                lattice(s_che_x1,s_che_y1,s_che_n1)=-lattice(s_che_x
1,s_che_y1,s_che_n1)
! !                write(*,*)'flipped spin',s_che_x1,s_che_y1,s_che_n
1,'back'
!             end if
!             s_che_counter1=s_che_counter1+1
!          end do
!          flooplength=loopspins-s_che_counter1+s_che_adj
!       end if

      end subroutine
```

```
      module defs

      implicit none
          !all the variables are here except those that are only used
 in a specific subroutine which may be defined there

      !*****DEFINED TYPES AND KINDS*****
      integer, parameter :: sp=selected_real_kind(6,37),dp=selected_r
eal_kind(15,307)
      type vector
         real(kind=dp),dimension(3)::components
      end type vector

      !*****GENERAL PURPOSE VARIABLES*****
      integer :: counter1,counter2,counter3,counter4,counter5,counter
6,counter7,counter8
      integer :: counter9,counter10,allocstat,holdup,called,errors
      integer :: cou11,cou12,cou13,cou14
      real :: starttime,stoptime
      real(kind=dp) :: random,random2,t_sign,f_sign
      integer, dimension(8) :: dtvalues
      character(8) :: date,time*10,label*3
      real(kind=dp), parameter :: pi=acos(-1._dp),r3=sqrt(3._dp),r2=s
qrt(2._dp),r6=sqrt(6._dp)
      real(kind=dp), parameter :: sperp=2._dp*r2/3._dp!magnitude of a
 unit length spin projected onto the kagome plane
      complex(kind=dp), parameter :: i=(0._dp,1._dp)

      !*****INPUT VARIABLES***** specified in the temp_start.sh file
      logical :: nmapping
      integer :: measurements,iterations,eqiterations,latt_start,side
      integer :: varyt_switch,varyf_switch,eqm_switch,mc_switch
      integer :: snapshots,loopsnapsteps
      real(kind=dp) :: j,mu,t_start,t_stop,minitemp_incr,bigtemp_incr
,sqtemp
      real(kind=dp) :: temp_start,temp_stop,field_start,field_stop,te
mp_incr,field_incr
      real(kind=dp), dimension(3) :: dir_start,dir_stop,field_para

      !*****MEASUREMENT AND STRUCTURE VARIABLES*****
      integer, dimension(:,:,:), allocatable :: lattice,flipcount
      real, dimension(:,:,:),allocatable :: avlattice
      real,dimension(:),allocatable::thetas
      real,dimension(1)::thetasmin
      integer :: n,shift
      real(kind=dp) :: magnetisation,e_old,delta_e,ener,stag_mag,ener
sq,magsq,sus,spheat,magx,magy,enertot
      real(kind=dp), dimension(3) :: v_magnetisation,field,v_stag_mag
,v_magsq
      real(kind=dp) :: temperature,f_strength,accept,tkast,magxtot,ma
gytot,magtot,fieldang,mang,magztot,momn,momnang

      !*****THEORETICAL CALCULATION VARIABLES******
      real(kind=dp) :: th_magx,th_magy,th_mag,th_sus,th_sph,th_e,th_m
ang
      real(kind=dp) :: th_mu1,th_mu2,th_mu3,th_alp2,th_alp3,th_z1,th_
z2,th_z3,th_da2dt,th_da3dt

      !*****LOOP VARIABLES*****
      integer :: loopstop,flooplength,defect,loop_x,loop_y,loop_n,fli
ptail,flipall
      integer :: loopspins,next_x,next_y,next_n,new_n,new_local,longl
oop,cla_x,cla_y,cla_n,clb_x,clb_y,clb_n
      integer,dimension(:),allocatable::loop_hist
      real(kind=dp)::loopaccept,eps0,eps1,pL,pR,pO,pLR,pLL,wO,wL,wR
```

```
      integer,dimension(:),allocatable::loop_xlist,loop_ylist,loop_nlist,looplength_freq

      !*****VECTOR VARIABLES*****
      !vectors are in the laboratory coordinate frame, x=left to right, y=bottom to top, z=out of plane
      !a1/2 and b1/2 are the basis and reciprocal vectors of the triangular lattice
      real(kind=dp), dimension(3), parameter :: a1=(/1._dp,0._dp,0._dp/)
      real(kind=dp), dimension(3), parameter :: a2=(/0.5_dp,r3/2._dp,0._dp/)
      real(kind=dp), dimension(3), parameter :: b1=2._dp*pi*(/1._dp,-1._dp/r3,0._dp/)
      real(kind=dp), dimension(3), parameter :: b2=2._dp*pi*(/0._dp,2._dp/r3,0._dp/)
      !All spins are defined positive pointing into the tetrahedron
      !A complete spin in the spin ice picture is formed from S_i+S_z so that it has length =1
      real(kind=dp), dimension(3), parameter :: S1=(/0._dp,-1._dp,0._dp/)*sperp
      real(kind=dp), dimension(3), parameter :: S2=(/(r3/2._dp),0.5_dp,0._dp/)*sperp
      real(kind=dp), dimension(3), parameter :: S3=(/-(r3/2._dp),0.5_dp,0._dp/)*sperp
      real(kind=dp), dimension(3), parameter :: Sz=(/0._dp,0._dp,1._dp/)*(1._dp/3._dp)
      real(kind=dp), dimension(3) :: ri
      contains

      !*****FUNCTIONS*****
      real(kind=dp) function kasttemp(f_kas_field,f_kas_theta)

      implicit none
          !this calculates the Kasteleyn temperature for the given field magnitude and angle
          !the angle is measured positive from [-1,-1,2] (y axis) clockwise to [-1,1,0] (x axis)

      real(kind=dp), intent(in) :: f_kas_field,f_kas_theta
      real(kind=dp) :: Tk1,deltaTk,Tk
      if(f_kas_field<0.000001)then
         kasttemp=0.0
      else
         deltaTk=1._dp
         Tk=0.2_dp
         do while(abs(deltaTk)>0.00000001) !this iterates to a value for Tk
            Tk1=(2._dp*sqrt(2._dp)*f_kas_field*cos(f_kas_theta))/(log(2._dp*cosh((2._dp*sqrt(2._dp)*f_kas_field*&
            sin(f_kas_theta))/(sqrt(3._dp)*Tk))))
            deltaTk=Tk1-Tk
            Tk=Tk1
         end do
         kasttemp=Tk1
      end if

      end function kasttemp

      !*************************************************************

      real(kind=dp) function fieldangle(f_fie_x,f_fie_y)
      !returns an angle from 0 to 2pi measured from the y axis in the clockwise sense given the x and y components
```

```
99        implicit none
100
101       real(kind=dp), intent(in):: f_fie_x,f_fie_y
102
103       fieldangle=0._dp
104       if(f_fie_x==0._dp.or.f_fie_y==0._dp)then
105          if(f_fie_x==0._dp.and.f_fie_y==0._dp)then !no field
106             fieldangle=0._dp
107          else
108             if(f_fie_x==0._dp)then
109                if(f_fie_y>0._dp)then
110                   fieldangle=0._dp
111                else
112                   fieldangle=pi
113                end if
114             elseif(f_fie_y==0._dp)then
115                if(f_fie_x>0._dp)then
116                   fieldangle=pi/2._dp
117                else
118                   fieldangle=(3._dp*pi)/2._dp
119                end if
120             end if
121          end if
122       elseif(f_fie_x>0._dp)then
123          if(f_fie_y>0._dp)then !in the first quadrant
124             fieldangle=atan(f_fie_x/f_fie_y)
125          else !in the second quadrant
126             fieldangle=atan(abs(f_fie_y/f_fie_x))+(pi/2._dp)
127          end if
128       elseif(f_fie_x<0._dp)then
129          if(f_fie_y>0._dp)then !in the fourth quadrant
130             fieldangle=atan(abs(f_fie_y/f_fie_x))+((3._dp*pi)/2._dp)
131          else !in the third quadrant
132             fieldangle=atan(abs(f_fie_x/f_fie_y))+pi
133          end if
134       end if
135
136       end function fieldangle
137
138       !*************************************************************
*****
139
140       real(kind=dp) function energy(f_ene_x,f_ene_y,f_ene_z) !returns
 the energy of a spin
141
142       implicit none
143
144       integer, intent(in) :: f_ene_x,f_ene_y,f_ene_z
145       real(kind=dp), dimension(3) :: f_ene_nn,f_ene_spin,f_ene_nnz,f_
ene_spinz
146       real(kind=dp) :: f_ene_temp
147
148       call neighbours(f_ene_x,f_ene_y,f_ene_z,f_ene_nn,0)
149       call spin(f_ene_x,f_ene_y,f_ene_z,f_ene_spin,0)
150       f_ene_temp=dot_product(f_ene_spin,(-j*f_ene_nn-mu*field))
151       call neighbours(f_ene_x,f_ene_y,f_ene_z,f_ene_nnz,1)
152       call spin(f_ene_x,f_ene_y,f_ene_z,f_ene_spinz,1)
153       energy=f_ene_temp+dot_product(f_ene_spinz,(-j*f_ene_nnz-field_p
ara))
154
155       end function energy
156
157
158       end module defs
```

```
      subroutine equilibrate

      use defs

      implicit none
          !this is called at each new temp or field to equilibrate th
e lattice, can use a fixed number of eq steps or other methods, see c
omments

      integer :: s_equi_counter1!,s_equi_defect,s_equi_counter2
!      real::s_equi_tempholder

      !commented sections allow for equilibration with temperature an
nealing and equilibration until defects have been removed, usually th
e minimal equilibration using a determined number of steps is enough.

!       s_equi_counter2=0
! 100   s_equi_defect=0
!       s_equi_tempholder=temperature
! !       temperature=0.5
!       do while(temperature-s_equi_tempholder>0.1)
!          write(*,*)'eq: eqiterations=',eqiterations
         do s_equi_counter1=1,eqiterations
            if(mc_switch.ne.2)then
               call montecarlo
            elseif(mc_switch.ne.1)then
               call loopmc
            end if
         end do
!          write(*,*)'eq: finished equilibrating'
!          temperature=0.5*temperature
!       end do
      accept=0.
      loopaccept=0.
      looplength_freq=0
!       temperature=s_equi_tempholder

      !the following section will keep equilibrating the lattice unti
l there are no defects, this may be commented out if not required
!       call defectcheck(0,s_equi_defect)
!       if(s_equi_defect==1)then
!       if(s_equi_defect==1.and.s_equi_counter2<=4)then
!          s_equi_counter2=s_equi_counter2+1
!          goto 100
!       elseif(s_equi_defect==1.and.s_equi_counter2>4)then
!          write(16,*)'Tried to anneal the defects from this lattice
5 times, giving up and carrying on with defects present.'
!       end if
!       write(*,*)'eq: exiting'
      end subroutine equilibrate
```

```
      subroutine equilibrium

      use defs

      implicit none

      !equilibrium program. only works with ssfs. shows how the energ
y and mag equilibrate as a fn of MC steps

      real(kind=dp), dimension(3) :: s_equ_spin
      integer :: s_equ_counter1,s_equ_counter2,s_equ_counter3,s_equ_c
ounter4,s_equ_counter5,s_equ_counter6,s_equ_counter7
      integer::s_equ_counter8
      real:: s_equ_rand1

      s_equ_counter8=0
      do s_equ_counter1=1,30!how many times to loop over the entire l
attice
         do s_equ_counter5=0,side
            do s_equ_counter6=0,side
               do s_equ_counter7=1,3
                  e_old=energy(s_equ_counter5,s_equ_counter6,s_equ_co
unter7)
                  delta_e=-2._dp*e_old
                  call random_number(s_equ_rand1)
                  if(delta_e<0._dp.or.s_equ_rand1<exp(-delta_e/temper
ature))then
                     lattice(s_equ_counter5,s_equ_counter6,s_equ_coun
ter7)=&
                     -lattice(s_equ_counter5,s_equ_counter6,s_equ_cou
nter7)
                  end if
                  s_equ_counter8=s_equ_counter8+1
                  ener=0._dp
                  magx=0._dp
                  magy=0._dp
                  do s_equ_counter2=0,side
                     do s_equ_counter3=0,side
                        do s_equ_counter4=1,3
                           call spin(s_equ_counter2,s_equ_counter3,s_
equ_counter4,s_equ_spin,0)
                           ener=ener+energy(s_equ_counter2,s_equ_coun
ter3,s_equ_counter4)
                           magx=magx+s_equ_spin(1)
                           magy=magy+s_equ_spin(2)
                        end do
                     end do
                  end do
                  write(15,"(I10,5x,3ES26.15E3)")s_equ_counter8,ener/(2.0
*n),(magx*9)/(n*4.*sqrt(2.)),(magy*9)/(n*4.*sqrt(2.))
               end do
            end do
         end do
!          call showlat
      end do
!       call showlat

      end subroutine equilibrium
```

```
      subroutine finalise

      use defs

      implicit none

      !write any final information to the information file
      call cpu_time(stoptime)
      write(17,*)'The simulation took ',(stoptime-starttime)/60.0,' minutes.'

      open(18,file='endconfigdata.out')!write the final lattice configurat
ion
      do counter1=0,side
         do counter2=0,side
            do counter3=1,3
               write(18,'(4I6)')counter1,counter2,counter3,lattice(cou
nter1,counter2,counter3)
            end do
         end do
      end do

      !close any open files, etc, tidy up
      close(12)
      close(13)
      close(14)
      close(15)
      close(16)
      close(17)
      close(18)
      close(19)
      close(20)
      close(21)
      close(22)
      close(39)

      end subroutine finalise
```

```
      subroutine initialise

      use defs

      implicit none
      !initialise anything that needs to only be done once at the beginning of the program
          !first read the temp_start.sh file (or field_start.sh etc) to determine the simulation parameters

      integer :: s_ini_counter3
      call init_random_seed
      read(*,*)j!interaction between spins
      read(*,*)mu
      read(*,*)(field_para(s_ini_counter3),s_ini_counter3=1,3)!field along [1,1,1] to simulate effects of fourth spin
      read(*,*)side!of the lattice, determines size of lattice
      read(*,*)varyt_switch!vary the temperature or
      read(*,*)varyf_switch!the field
      read(*,*)temp_start
      read(*,*)field_start
      read(*,*)(dir_start(s_ini_counter3),s_ini_counter3=1,3)!starting direction of the field
      if(varyt_switch==1)then!temperature can change in big and small increments to allow fine measurement over an important region
         read(*,*)temp_stop
         if(temp_stop>=tiny(0._dp))then
            temp_stop=temp_stop-tiny(0._dp)
         end if
         if(temp_start-temp_stop>0._dp)then!temperature is going down
            t_sign=1._dp
         else!temperature is going up
            t_sign=-1._dp
         end if
         read(*,*)bigtemp_incr
         read(*,*)t_start
         read(*,*)t_stop
         read(*,*)minitemp_incr
      elseif(varyt_switch/=1)then
         temp_stop=temp_start+0.1 !otherwise the temp loop will finish immediately
         t_sign=-1._dp
      end if
      if(varyf_switch==1)then
         read(*,*)field_stop
         if(field_stop>=0.00001)then
            field_stop=field_stop-0.00001
         end if
         if(field_start-field_stop>0._dp)then
            f_sign=1._dp
         else
            f_sign=-1._dp
         end if
         read(*,*)field_incr
      elseif(varyf_switch/=1)then
         field_stop=field_start+0.1 !otherwise the field loop will finish immediately
         f_sign=-1._dp
      end if
      read(*,*)eqm_switch!whether to perform an equilibration simulation, ie measure how the system equilibrates
      read(*,*)latt_start!determine the initial lattice config
      read(*,*)measurements!how many measurements at each temp/field
      read(*,*)iterations!how many MC steps between measuring
      read(*,*)eqiterations!how many MC steps to equilibrate the latt
```

```
ice after changing the temp/field
      read(*,*)mc_switch!choose single spin flip or loop flip MC step
s
      read(*,*)longloop!choose to close loops with long only or short
 and long condition
      if(longloop==0)then
         write(*,*)'The loop currently cant handle short loop closing!'
         stop
      end if
      read(*,*)nmapping!whether to produce neutron scattering lattice
 config files
      read(*,*)sqtemp!if so at what temperature
      read(*,*)loopsnapsteps!how many loop MC steps between NS lattic
e snapshots
      read(*,*)snapshots!how many snapshots to take, for s.s.f. MC th
is is how many times to reset the sim
      !initialise all possible variables etc
      if(sqrt(dir_start(1)**2+dir_start(2)**2+dir_start(3)**2)>0.0000
00001)then!normalise the field direction
         dir_start=(1._dp/sqrt(dir_start(1)**2+dir_start(2)**2+dir_st
art(3)**2))*dir_start
      end if
      called=0
      cou14=0
      n=(side+1)*(side+1)*3!the number of spins in the lattice
      allocate(lattice(0:side,0:side,3),loop_xlist(0:10*n),loop_ylist
(0:10*n),loop_nlist(0:10*n),looplength_freq(0:10*n),&
      flipcount(0:side,0:side,3),avlattice(0:side,0:side,3),thetas(0:
side),loop_hist(1:10*n),stat=allocstat)
      if(allocstat>0)then
         stop
      end if
      flipall=0
      loop_hist=0
      flooplength=0
      loopspins=0
      loop_xlist=0
      loop_ylist=0
      loop_nlist=0
      counter6=0
      temperature=temp_start
      f_strength=field_start
      field=f_strength*dir_start
      temp_incr=0._dp
      magnetisation=0._dp
      v_magnetisation=0._dp
      ener=0._dp
      accept=0.
      stag_mag=0._dp
      v_stag_mag=0._dp
      loopaccept=0.
      looplength_freq=0
      counter7=0 !this is a counter for the number of times show latt
ice is called
      counter8=0 !this is a counter for the number of times av lattic
e is called
      enersq=0._dp
      magsq=0._dp
      tkast=kasttemp(sqrt(field(1)**2+field(2)**2+field(3)**2),atan(f
ield(1)/field(2)))
      magx=0._dp
      magy=0._dp
      flipcount=0
      avlattice=0.0
      th_mag=0._dp
```

```
      fliptail=0
      th_sus=0._dp
      magytot=0._dp
      magxtot=0._dp
      magztot=0._dp
          !when theta=0 the loop should go vertically so the pbc must be shifted by half the lattice edge length.
          !calculate the field angle which will range from 0 to 2pi where 2pi is called 0 again to check it is in the correct region.
      fieldang=fieldangle(field(1),field(2))
      !if the field angle is within -pi/3 of the y axis then reflect it to a positive value. this might work, everything is designed around the field being between 0 and pi/3 so choose this to be safe!
      if(fieldang>(pi/3._dp))then
         if(fieldang>((5._dp*pi)/3._dp))then !the loops only work if the field is +-pi/3 of the y axis, if its -ve then mirror the angle
            fieldang=(2._dp*pi)-fieldang
            write(*,*)'WARNING - the field angle is between 0 and -pi/3 of the y-axis.'
         else
            write(*,*)'ERROR - the field angle is not within +-pi/3 of the y-axis.'
            stop
         end if
      end if
      shift=((side+1)/2)!this makes loops that are parallel to -1-12 connect through the vertical PB
          !this shifts the vertical PBC by half the lattice width 'twisting' the lattice so the connectivity is that of a square not a lozenge

      !initialise the lattice
      call latt_init(latt_start)

      !open any necessary files to write data or info to
      call date_and_time(DATE=date,TIME=time,VALUES=dtvalues)
      !--------open the general files used in every simulation-------
      open(17,file='info_'//date//time(1:6)//'.out',status='new')
      if(eqm_switch==1)then
         open(15,file='eqm_'//date//time(1:6)//'data.out',status='new')
      else
         if(varyt_switch==1)then
            open(12,file='t_'//date//time(1:6)//'data.out',status='new')
         elseif(varyf_switch==1)then
            open(13,file='f_'//date//time(1:6)//'data.out',status='new')
         end if
         open(16,file='progress',status='new')
         open(19,file='looplengths_'//date//time(1:6)//'.out',status='new')
         open(20,file='probabilitiesdata.out',status='new')
      end if
      !--------write some information about the simulation-------
      write(17,*)'Information file for a simulation on a Kagome layer of the spin ice lattice.'
      write(17,*)
      write(17,*)'This simulation is expected to show a Kasteleyn transition at a temperature of ',tkast
      write(17,*)
      write(17,'(A30,I2,A1,I2,A1,I4,A4,I2,A1,I2,A1,I2)')'The simulation was started on ',dtvalues(3),'/',dtvalues(2),'/',&
      dtvalues(1),' at ',dtvalues(5),':',dtvalues(6),':',dtvalues(7)
      write(17,*)
      write(17,*)'These were the input parameters'
      write(17,*)
      write(17,'(A30,I15)')'No. of spins in lattice:',n
      write(17,'(A30,I15)')'Measurements per temp:',measurements
      write(17,'(A30,I15)')'MC steps between meas:',iterations
```

```
163         write(17,'(A30,I15)')'Equil. MC steps per temp:',eqiterations
164         write(17,*)
165         write(17,'(A30)')'-- Temperature parameters --'
166         if(varyt_switch==1)then
167            write(17,'(A30,ES12.3)')'Starting temperature:',temp_start
168            write(17,'(A30,ES12.3)')'Finishing temperature:',temp_stop+0.0001
169            write(17,'(A30,ES12.3)')'Big temperature step size:',bigtemp_incr
170            write(17,'(A30,ES12.3)')'Fine temp step zone: high end',t_stop
171            write(17,'(A30,ES12.3)')'Fine temp step zone: low end',t_start
172            write(17,'(A30,ES12.3)')'Small temperature step size:',minitemp_incr
173         else
174            write(17,'(A30,ES12.3)')'Temperature of sim.',temperature
175         end if
176         write(17,*)
177         write(17,'(A30)')'-- Field parameters --'
178         write(17,'(A30,3ES12.3)')'Field vector (norm):',dir_start
179         write(17,'(A30,ES12.3)')'Initial field magnitude:',field_start
180         if(varyt_switch==1)then
181            write(17,'(A30,ES12.3)')'Final field magnitude:',field_stop-0.1
182         else
183            write(17,'(A30,ES12.3)')'Final field magnitude:',field_stop+0.00001
184         end if
185         if(fieldang>(pi/3._dp).and.fieldang<((5._dp*pi)/3._dp))then
186            write(17,'(A55)')'WARNING - field direction not compatible with loop MC!'
187         end if
188         write(17,*)
189         if(nmapping.eqv..true.)then
190            write(17,*)'This run will produce spin configuration files for a simulated S(Q) map
    '
191            write(17,'(A30,ES12.3)')'Temperature of S(Q) simulation:',sqtemp
192            write(17,'(A30,I10)')'No. of independant snapshots/files produced:',snapshot
    s
193         elseif(nmapping.eqv..false.)then
194            write(17,*)'This run will not produce data for a simulated S(Q) map'
195         end if
196         select case(mc_switch)
197            case(1)
198               write(17,*)'This simulation used only single MC steps'
199            case(2)
200               write(17,*)'This simulation used only loop move MC steps'
201            case(3)
202               write(17,*)'This simulation used single and loop move MC steps'
203         end select
204         select case(latt_start)
205            case(1)
206               write(17,*)'The lattice was initialised randomly'
207            case(2)
208               write(17,*)'The lattice was initialised in a q=x state.'
209            case(3)
210               write(17,*)'The lattice was initialised in a q=0 state, s1 out, s2 and s3 in.'
211            case(4)
212               write(17,*)'The lattice was initialised in a random and "half" constrained state.
     2 in 2 out on every up tetrahedron.'
213            case(5)
214               write(17,*)'The lattice was initialised from the configuration in the file startco
    nfig.'
215            case(6)
216               write(17,*)'The lattice was initialised ferromagnetically, spins 2 and 3 were pa
    rallel to spin 1.'
217            case(7)
218               write(17,*)'The lattice was initialised in a q=0 state, all spins out.'
219            case(8)
220               write(17,*)'The lattice was initialised randomly with zeros around the edge to
    simulate no PBC.'
221            case(9)
```

```
222             write(17,*)'The lattice was initialised as a triangular lattice, s2=s2=0.'
223         end select
224         write(17,*)
225 
226         end subroutine initialise
```

```
1         subroutine init_random_seed()
2
3         implicit none
4             !initialise the random seed for the random number generator
    from the system clock
5
6         integer :: s_ini_counter1, s_ini_clock, s_ini_size
7         integer, dimension(30) :: s_ini_seed
8
9         s_ini_size=30
10        call random_seed(size=s_ini_size)
11        call system_clock(count=s_ini_clock)
12        s_ini_seed=s_ini_clock+37*(/(s_ini_counter1-1,s_ini_counter1=1,
    30)/)
13        call random_seed(put=s_ini_seed)
14
15        end subroutine
```

```
      subroutine latt_init(s_lat_start)

      use defs

      implicit none
      !initialises the lattice in a configuration determined by the n
  umber passed to the subroutine

      integer,intent(in)::s_lat_start
      integer::s_lat_counter1,s_lat_counter2,s_lat_counter3,temp1,tem
  p2,temp3


      select case (s_lat_start)
         case(1)!completely random
            do s_lat_counter1=0,side
               do s_lat_counter2=0,side
                  call random_number(random)
                  if(random>0.5_dp)then
                     lattice(s_lat_counter1,s_lat_counter2,1)=1
                  else
                     lattice(s_lat_counter1,s_lat_counter2,1)=-1
                  end if
                  call random_number(random)
                  if(random>0.5_dp)then
                     lattice(s_lat_counter1,s_lat_counter2,2)=1
                  else
                     lattice(s_lat_counter1,s_lat_counter2,2)=-1
                  end if
                  call random_number(random)
                  if(random>0.5_dp)then
                     lattice(s_lat_counter1,s_lat_counter2,3)=1
                  else
                     lattice(s_lat_counter1,s_lat_counter2,3)=-1
                  end if
               end do
            end do
         case(2)!spin 1 in, then either 2 in 3 out, or 2 out 3 in
            lattice=1
            lattice(:,:,3)=-1
            do s_lat_counter1=0,side
               call random_number(random)
               if(random>0.5_dp)then
                  lattice(:,s_lat_counter1,2)=-1
                  lattice(:,s_lat_counter1,3)=1
               end if
            end do
         case(3)!q=0, spin1 out, spin 2 and 3 in
            lattice(:,:,1)=-1
            lattice(:,:,2)=1
            lattice(:,:,3)=1
         case(4)!sqrt3xsqrt3 config
            lattice=1
            do s_lat_counter1=0,side
               do s_lat_counter2=0,side
                  select case(mod(s_lat_counter1,3))
                     case(0)
                        select case(mod(s_lat_counter2,3))
                           case(0)
                              lattice(s_lat_counter1,s_lat_counter2,1
  )=-1
                           case(1)
                              lattice(s_lat_counter1,s_lat_counter2,3
  )=-1
                           case(2)
```

```
                                    lattice(s_lat_counter1,s_lat_counter2,2
)=-1
                              end select
                           case(1)
                              select case(mod(s_lat_counter2,3))
                                 case(0)
                                    lattice(s_lat_counter1,s_lat_counter2,2
)=-1
                                 case(1)
                                    lattice(s_lat_counter1,s_lat_counter2,1
)=-1
                                 case(2)
                                    lattice(s_lat_counter1,s_lat_counter2,3
)=-1
                              end select
                           case(2)
                              select case(mod(s_lat_counter2,3))
                                 case(0)
                                    lattice(s_lat_counter1,s_lat_counter2,3
)=-1
                                 case(1)
                                    lattice(s_lat_counter1,s_lat_counter2,2
)=-1
                                 case(2)
                                    lattice(s_lat_counter1,s_lat_counter2,1
)=-1
                              end select
                        end select
                     end do
                  end do
               case(5)!read config from file
                  open(20,file='startconfig.data')
                  do s_lat_counter1=0,side
                     do s_lat_counter2=0,side
                        do s_lat_counter3=1,3
                           read(20,'(4I6)')temp1,temp2,temp3,lattice(s_lat_c
ounter1,s_lat_counter2,s_lat_counter3)
                        end do
                     end do
                  end do
                  close(20)
               case(6)!special set up for testing s(q) map. also changes al
l spins to spin 1, hence completely ferromagnetic lattice
                  lattice=1
               case(7)!all spins out
                  lattice=-1
               case(8)!random with zeros around the edge to sim no PBC's
                  do s_lat_counter1=0,side
                     do s_lat_counter2=0,side
                        call random_number(random)
                        if(random>0.5_dp)then
                           lattice(s_lat_counter1,s_lat_counter2,1)=1
                        else
                           lattice(s_lat_counter1,s_lat_counter2,1)=-1
                        end if
                        call random_number(random)
                        if(random>0.5_dp)then
                           lattice(s_lat_counter1,s_lat_counter2,2)=1
                        else
                           lattice(s_lat_counter1,s_lat_counter2,2)=-1
                        end if
                        call random_number(random)
                        if(random>0.5_dp)then
                           lattice(s_lat_counter1,s_lat_counter2,3)=1
                        else
```

```
                    lattice(s_lat_counter1,s_lat_counter2,3)=-1
                 end if
              end do
           end do
           do s_lat_counter1=0,side
              lattice(0,s_lat_counter1,:)=0
              lattice(side,s_lat_counter1,:)=0
              lattice(s_lat_counter1,0,:)=0
              lattice(s_lat_counter1,side,:)=0
           end do
        case(9)!only spin 1 is initialised, makes a triangular latti
ce
           lattice=0
           lattice(:,:,1)=1
     end select

     end subroutine
```

```
      subroutine loopflip

      use defs

      implicit none
      !this sub actually flips the spin and updates the number of spi
ns in the loop and the list of their x y and n coords
      !add the current spin to the loop
      loopspins=loopspins+1!number of spins in the loop, references t
he spin within the loop
      loop_xlist(loopspins)=loop_x!x pos of that spin
      loop_ylist(loopspins)=loop_y
      loop_nlist(loopspins)=loop_n!n=1 2 or 3 for each x,y pair
      loop_hist((3*(side+1)*loop_x)+(3*loop_y)+loop_n)=1!loop history
, initially all zero, as the loop passes over spins it changes the va
lue to 1
      lattice(loop_x,loop_y,loop_n)=-lattice(loop_x,loop_y,loop_n)

      end subroutine
```

```
      subroutine loopmc

      use defs

      implicit none
          ! this subroutine is the loop algorithm update step, each time it is called it will (potentially) add one loop to the lattice
      !The loops only work above Tk as the probabilities may be P>1 or P<0 below this temp which would require backtracking moves so check for this and skip the process if necessary
      if(temperature<tkast)then
         goto 100
      end if

      !-----INITIALISATION OF THE LOOP-------
      !initialise/reset variables
      loopstop=0;defect=0;fliptail=0;loop_xlist=0;loop_ylist=0;loop_nlist=0;loopspins=0;loop_hist=0;flipall=0
      call init_random_seed
      !randomly choose a spin coming out of an up triangle, loop_n determines what type of up tri it is. This positions the defect/head of the string in a down triangle, it must then jump to a neighbouring down triangle in every move. This will be the first spin in the loop.
      call random_number(random)
      loop_x=nint(random*side)
      call init_random_seed
      call random_number(random)
      loop_y=nint(random*side)
      if(lattice(loop_x,loop_y,1)<0)then
         loop_n=1
         if(lattice(loop_x,loop_y,2)<0.or.lattice(loop_x,loop_y,3)<0) then
            loop_n=0
            write(16,*)'ERROR - the loop started at a defect, 1 out AND 2/3 out'
         endif
         cla_n=2
         clb_n=3
      elseif(lattice(loop_x,loop_y,2)<0)then
         loop_n=2
         if(lattice(loop_x,loop_y,3)<0)then
            loop_n=0
            write(16,*)'ERROR - the loop started at a defect, 2 out AND 3 out'
         end if
         cla_n=3
         clb_n=1
      elseif(lattice(loop_x,loop_y,3)<0)then
         loop_n=3
         cla_n=1
         clb_n=2
      else
         loop_n=0
         write(16,*)'ERROR - the loop started at a defect, all in?'
      end if
      if(loop_n==0)then
         call showlat
         write(16,'(A25,3I3)')'defect at ',loop_x,loop_y,loop_n
         return
      end if

      !calculate the two spins that are on the same down triangle as the first to use for closing check. one of these must be the last in the loop
      call next(loop_x,loop_y,loop_n,cla_n,0,cla_x,cla_y,cla_n)
      call next(loop_x,loop_y,loop_n,clb_n,0,clb_x,clb_y,clb_n)
```

```
      !if loop_n=1 it is a slave spin and we must flip another spin t
  o get to the next down triangle.
      if(loop_n==1)then
         call loopflip
         call random_number(random)
         if(random<=pR)then
            call next(loop_x,loop_y,loop_n,2,1,next_x,next_y,next_n)
         else
            call next(loop_x,loop_y,loop_n,3,1,next_x,next_y,next_n)
         end if
         loop_x=next_x
         loop_y=next_y
         loop_n=next_n
         call loopflip
      end if
      !-----END INITIALISATION OF THE LOOP---

      !-----CYCLE TO BUILD THE LOOP----------
      do while(loopstop==0.and.defect==0)
         call loopmove
         call check
      end do
      !-----CYCLE TO BUILD THE LOOP---------

      !-----AFTER CLOSING A LOOP--------------
      !either the loop closed or it hit a defect
      if(loopstop==1)then
         loopaccept=loopaccept+1.
         !put the looplength into the cumulative frequency tally
         looplength_freq(flooplength)=looplength_freq(flooplength)+1
      elseif(defect==1)then
         !transport the defect across the lattice - wormhole
      end if
      !-----AFTER CLOSING A LOOP-----------------

100   end subroutine loopmc
```

```
      subroutine loopmove

      use defs

      implicit none
      ! this sub works out what the next spin in the loop should be as it is being constructed
      ! the numbers in the comments refer to the vertex type (1 to 5) illustrated in the thesis

      ! the spin loop_n is the last spin that was flipped
      new_n=0

      if(loop_n==1)then !a string is being removed from the lattice, we must find out which way it goes out of the down tri and follow it.
         call next(loop_x,loop_y,loop_n,2,0,next_x,next_y,next_n)
         call random_number(random)
         if(lattice(next_x,next_y,next_n)<0)then!we must leave though spin 2 and we complete a 3 to 1 or a 5 to 1
            loop_x=next_x
            loop_y=next_y
            loop_n=next_n
            call loopflip
            if(random<=p0)then
               new_n=1
            else
               new_n=3
            end if
         else! we leave through spin 3 and complete a 2 to 1 or a 4 to 1
            call next(loop_x,loop_y,loop_n,3,0,next_x,next_y,next_n)
            loop_x=next_x
            loop_y=next_y
            loop_n=next_n
            call loopflip
            if(random<=p0)then
               new_n=1
            else
               new_n=2
            end if
         end if
         new_local=1
      elseif(loop_n==2)then !the loop is in the down tri of a v1, v2 or v4
         call next(loop_x,loop_y,loop_n,1,0,next_x,next_y,next_n)
         if(lattice(next_x,next_y,next_n)>0)then!you are in a 2 or 4
            if(loop_nlist(loopspins)==2)then !no choice but to exit through spin 3, update the head of the loop without flipping spin 3, that will happen in the next move
               call next(loop_x,loop_y,loop_n,3,0,next_x,next_y,next_n)
               loop_x=next_x
               loop_y=next_y
               loop_n=next_n
            else !last spin was a 3 so you're in the down tri below a v2 or v3
               call loopflip !first you must flip spin 2, then choose LO or LR
               call random_number(random)
               if(random<=p0)then
                  new_n=1
               else
                  new_n=3
               end if
               new_local=1
```

```
            end if
         else! v1, flip the slave automatically and choose left or ri
ght
            loop_x=next_x
            loop_y=next_y
            loop_n=next_n
            call loopflip
            call random_number(random)
            if(random<=pL)then
               new_n=3
            else
               new_n=2
            end if
            new_local=1
         end if
      elseif(loop_n==3)then ! the loop is in the down tri of a v1, v3
 or v5
         call next(loop_x,loop_y,loop_n,1,0,next_x,next_y,next_n)
         if(lattice(next_x,next_y,next_n)>0)then!you are in a 3 or 5,

            if(loop_nlist(loopspins)==3)then !no choice but to exit t
hrough spin 2, update the head of the loop without flipping spin 2, t
hat will happen in the next move
               call next(loop_x,loop_y,loop_n,2,0,next_x,next_y,next_
n)
               loop_x=next_x
               loop_y=next_y
               loop_n=next_n
            else !last spin was a 2 so you're in the down tri below a
 v4 or v5
               call loopflip !first you must flip spin 3, then choose L
O or LR
               call random_number(random)
               if(random<=pO)then
                  new_n=1
               else
                  new_n=2
               end if
               new_local=1
            end if
         else! v1, flip the slave automatically and choose left or ri
ght
            loop_x=next_x
            loop_y=next_y
            loop_n=next_n
            call loopflip
            call random_number(random)
            if(random<=pL)then
               new_n=3
            else
               new_n=2
            end if
            new_local=1
         end if
      end if

      if(new_n/=0)then
         call next(loop_x,loop_y,loop_n,new_n,new_local,next_x,next_y
,next_n)
         loop_x=next_x
         loop_y=next_y
         loop_n=next_n
         call loopflip
      end if
```

```
110         end subroutine
```

```
      program spinice

      !*****NOTES ON THE 'SPINICE' PROGRAM*****
      !1. all the files produced are labelled '[code][dateandtime]' with the following codes
      !   eqm_ = data from an equilibrium run to show how the system equilibrates as a func of MC steps
      !   t_ = data from a run with temperature as the variable parameter
      !   f_ = data from a run with field strength as the variable parameter
      !2. the lattice is defined as a hexagonal bravais lattice with a triangular decoration to give the kagome lattice.
      !   lattice points are defined as +-1 then multiplied by a vector and a magnitude to represent the x and y or the z part
      !   of the 'real' spin. in the x and y case the magnitude is (2*sqrt(2))/3.0 and in the z case it is 1./3. (see the spins defined in the defs module)
      !   the point (0,0,0) is in the bottom left corner. this is only relevant for the lattice plotting routine, showlat
      !3. the triangles are 'up' triangles with the 4th tetrahedral spin in the center pointing 'out' along the positive
      !   z-axis, above the kagome plane.
      !   the down triangles are defined between the 'up' triangles and have their 4th spin pointing 'in' to the triangle from 'below'.
      !4. if a spin is pointing 'into' a triangle it is + in the lattice, pointing 'out' is -.
      !5. all vector quantities, like applied field etc. are defined as (x,y,z) where z is the [111] cubic coordinate
      !   direction and is perpendicular to the kagome plane, x and y are orthogonal and x runs left to right
      !6. the lattice vectors are defined as a and b for the rhombohedral unit cell.
      !7. writing 'call showlat' at any point will produce an output file with a snapshot of the lattice at that time.
      !   it can be plotted with the lattplot.sh file, or gnuplot with vectors.

      !all subroutines have a small description at the top of the file, there are (hopefully) lots of comments thoughout as well

      use defs

      implicit none

      call cpu_time(starttime) !to measure how long the complete sim takes
      call initialise !set everything up
      if(eqm_switch==1)then
         call equilibrium
         stop
      end if

100 temp: do while((temperature-temp_stop)*t_sign>=1E-8) !while temperature is not at the final value...
fld:     do while((f_strength-field_stop)*f_sign>=1E-8) !while field is not at the final value...
            call bweights !calculate boltzmann weights for the loop algorithm
            call equilibrate !equilibrate the lattice
meas:       do counter4=1,measurements !loop over measurements
iter:          do counter5=1,iterations !it is possible to do more than one MC step between measurements
                  if(mc_switch.ne.2)then
                     call montecarlo !this is the single spin flip (st
```

```
andard) MC algorithm
                  elseif(mc_switch.ne.1)then
                     call loopmc !this is the loop update algorithm
                  end if
               end do iter
               call measure !measure the properties of the lattice
            end do meas
            if(nmapping.eqv..true.)then!produce a file of the spin po
sition reference and sign for making neutron scattering maps with the
 scatter program
               if(abs(temperature-sqtemp)<=1E-8)then !only do this if
 the sim is at the reqd temp for the snapshot
                  cou14=cou14+1
                  write(16,*)'Writing spin configuration file',cou14,'of',snapsh
ots
                  write(16,*)
                  if(cou14<10)then
                     write(label,'(I1,I1,I1)')0,0,cou14
                  elseif(cou14>9.and.cou14<100)then
                     write(label,'(I1,I2)')0,cou14
                  else
                     write(label,'(I3)')cou14
                  end if
                  open(87,file='spin_config'//label//'data.out',status='new
')
                  do cou11=0,side
                     do cou12=0,side
                        do cou13=1,3
                           write(87,'(4I6)')cou11,cou12,cou13,lattice(
cou11,cou12,cou13)
                        end do
                     end do
                  end do
                  close(87)
                  call flush
                  called=called+1 !this records how many times the sq
 map has been produced
               end if
            end if
            if(cou14<2)then
               call output !this outputs the thermo measurements, this
 if stops that happening repeatedly at the same temp when the sq maps
 are being written
            end if
            if(varyf_switch==1)then !if the field stre
               f_strength=f_strength+field_incr
               field=f_strength*dir_start
               tkast=kasttemp(sqrt(field(1)**2+field(2)**2+field(3)**
2),atan(field(1)/field(2)))
            else
               exit
            end if
         end do fld
         if(varyt_switch==1)then!update the temperature using either
big or small temp steps and if the temp will cross into a
            temp_incr=bigtemp_incr!different step size regime then ju
st update the temp to the nearest point to the start of the new bit.
            if(t_sign>0.5)then !temperature is decreasing
               if(temperature-t_stop<=1E-6)then
                  if(temperature-t_start>=1E-6)then
                     temp_incr=minitemp_incr
                  end if
               elseif(temperature+bigtemp_incr<t_stop)then
                  temp_incr=-(temperature-t_stop)
               end if
```

```
95              elseif(t_sign<-0.5)then !temperature is increasing
96                 if(t_start-temperature<=1E-6)then
97                    if(t_stop-temperature>=1E-6)then
98                       temp_incr=minitemp_incr
99                    end if
100                elseif(temperature+bigtemp_incr>t_start)then
101                   temp_incr=t_start-temperature
102                end if
103             end if
104             temperature=temperature+temp_incr
105          elseif(varyf_switch==1)then
106             temperature=temp_stop
107          end if
108       end do temp
109
110       if(nmapping.eqv..true.)then!if the simulation is producing spin
 configuration files for neutron scattering then reset (in the approp
riate way for ssf or loops) to make the next file.
111          if(snapshots>called)then
112             if(mc_switch==1)then!using ssf
113                call latt_init(latt_start)
114                temperature=temp_start
115             else!using loops
116                if(called>0)then!taking snaps has started, reset the t
emperature
117                   temperature=temperature-temp_incr
118                end if
119                measurements=loopsnapsteps
120             end if
121             goto 100
122          end if
123       end if
124
125       call finalise
126
127       end program spinice
```

```
      subroutine measure

      use defs

      implicit none
      !measures the properties of the lattice after x MC updates

      real(kind=dp), dimension(3) :: s_mea_spin
      integer :: s_mea_counter1,s_mea_counter2,s_mea_counter3

      ener=0._dp
      magx=0._dp
      magy=0._dp
      do s_mea_counter1=0,side
         do s_mea_counter2=0,side
            do s_mea_counter3=1,3
               !---------ENERGY----------------
               ener=ener+energy(s_mea_counter1,s_mea_counter2,s_mea_c
ounter3)
               !---------TOTAL MAGNETISATION----
               call spin(s_mea_counter1,s_mea_counter2,s_mea_counter3
,s_mea_spin,0)
               magx=magx+s_mea_spin(1)
               magy=magy+s_mea_spin(2)
            end do
         end do
      end do
      !take running totals over the configuration
      magxtot=magxtot+magx
      magytot=magytot+magy
      magztot=magztot+n*(4._dp/9._dp)
      magtot=magtot+sqrt(magx**2+magy**2)
      magsq=magsq+magx**2+magy**2
      enersq=enersq+ener**2
      enertot=enertot+ener

      end subroutine measure
```

```
      subroutine montecarlo

      use defs

      implicit none
          !this is the standard monte carlo single spin flip update algorithm

      integer :: s_mon_counter1,s_mon_counter2,s_mon_counter3
      real(kind=dp) :: s_mon_rand1

      do s_mon_counter1=0,side
         do s_mon_counter2=0,side
            do s_mon_counter3=1,3
               e_old=energy(s_mon_counter1,s_mon_counter2,s_mon_counter3)
               delta_e=-2._dp*e_old !this works as the update can only flip a spin so Enew=-Eold
               call random_number(s_mon_rand1)
               if(delta_e<0._dp.or.s_mon_rand1<exp(-delta_e/temperature))then
                  lattice(s_mon_counter1,s_mon_counter2,s_mon_counter3)=&
                  -lattice(s_mon_counter1,s_mon_counter2,s_mon_counter3)
                  accept=accept+1._dp
               end if
            end do
         end do
      end do

      end subroutine montecarlo
```

```
      subroutine neighbours(s_nei_row,s_nei_col,s_nei_num,s_nei_nn,s_nei_zswitch)

      !this subroutine returns a 3-component vector which contains the sum of the lattice point multiplied by its corresponding
          !spin vector for the row, col, and num of the four neighbours to the coordinates of the spin entered as the argument.
          !neighbours are numbered starting at 1 with the one directly above (in the y/col direction) or the closest to that
          !moving clockwise, then proceeding clockwise.

      use defs, only: side,lattice,S1,S2,S3,Sz,dp

      implicit none

      integer, intent(in) :: s_nei_row,s_nei_col,s_nei_num,s_nei_zswitch
      integer, dimension(3) :: s_nei_nn1,s_nei_nn2,s_nei_nn3,s_nei_nn4
      real(kind=dp), dimension(3), intent(out) :: s_nei_nn

      select case(s_nei_num)
         case(1)
            call next(s_nei_row,s_nei_col,s_nei_num,2,0,s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))
            call next(s_nei_row,s_nei_col,s_nei_num,3,1,s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))
            call next(s_nei_row,s_nei_col,s_nei_num,2,1,s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))
            call next(s_nei_row,s_nei_col,s_nei_num,3,0,s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3))
            if(s_nei_zswitch==1)then
               s_nei_nn=(lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))+&
               lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))+&
               lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))+&
               lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3)))*Sz
            else
               s_nei_nn=lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))*S2+&
               lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))*S3+&
               lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))*S2+&
               lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3))*S3
            end if
         case(2)
            call next(s_nei_row,s_nei_col,s_nei_num,1,1,s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))
            call next(s_nei_row,s_nei_col,s_nei_num,3,1,s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))
            call next(s_nei_row,s_nei_col,s_nei_num,1,0,s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))
            call next(s_nei_row,s_nei_col,s_nei_num,3,0,s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3))
            if(s_nei_zswitch==1)then
               s_nei_nn=(lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))+&
               lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))+&
               lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))+&
               lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3)))*Sz
            else
               s_nei_nn=lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3))*S1+&
               lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))*S3+&
               lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))*S1+&
               lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3))*S3
```

```
48              end if
49           case(3)
50              call next(s_nei_row,s_nei_col,s_nei_num,2,0,s_nei_nn1(1),
    s_nei_nn1(2),s_nei_nn1(3))
51              call next(s_nei_row,s_nei_col,s_nei_num,1,0,s_nei_nn2(1),
    s_nei_nn2(2),s_nei_nn2(3))
52              call next(s_nei_row,s_nei_col,s_nei_num,2,1,s_nei_nn3(1),
    s_nei_nn3(2),s_nei_nn3(3))
53              call next(s_nei_row,s_nei_col,s_nei_num,1,1,s_nei_nn4(1),
    s_nei_nn4(2),s_nei_nn4(3))
54              if(s_nei_zswitch==1)then
55                 s_nei_nn=(lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(
    3))+&
56                 lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))+&
57                 lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))+&
58                 lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3)))*Sz
59              else
60                 s_nei_nn=lattice(s_nei_nn1(1),s_nei_nn1(2),s_nei_nn1(3
    ))*S2+&
61                 lattice(s_nei_nn2(1),s_nei_nn2(2),s_nei_nn2(3))*S1+&
62                 lattice(s_nei_nn3(1),s_nei_nn3(2),s_nei_nn3(3))*S2+&
63                 lattice(s_nei_nn4(1),s_nei_nn4(2),s_nei_nn4(3))*S1
64              end if
65        end select
66
67        end subroutine neighbours
```

```
      subroutine next(s_nex_xin,s_nex_yin,s_nex_nin,s_nex_nnew,s_nex_
local,s_nex_xout,s_nex_yout,s_nex_nout)
      !this calculates the neighbouring spin given the current spin c
oords and the type (1 2 or 3) and whether it is in the same triangle
          !(local) of the next spin
          !x,y,nin are the reference for the current spin; x,y,nout a
re the reference for the next spin
      !nnew is the type of spin the next one should be (1,2 or 3)
      !local indicates whether the next spin should be on the same (1
) or neighbouring (0) triangle to the current spin
          !this is where the shifted PBC are created
      use defs

      implicit none

      integer,intent(in)::s_nex_xin,s_nex_yin,s_nex_nin,s_nex_nnew,s_
nex_local
      integer,intent(out)::s_nex_xout,s_nex_yout,s_nex_nout

      if(s_nex_nin==s_nex_nnew)then!sub is asked to go to the spin it
's already on
         s_nex_xout=s_nex_xin
         s_nex_yout=s_nex_yin
         s_nex_nout=s_nex_nin
         return
      end if

      select case(s_nex_nin)
         case(1)
            if(s_nex_nnew==2)then
               if(s_nex_local==0)then
                  s_nex_xout=s_nex_xin
                  s_nex_yout=s_nex_yin+1
                  s_nex_nout=2
                  if(s_nex_yout>side)then
                     s_nex_yout=0
                     s_nex_xout=modulo(s_nex_xout+shift,side+1)
                  end if
               elseif(s_nex_local==1)then
                  s_nex_xout=s_nex_xin
                  s_nex_yout=s_nex_yin
                  s_nex_nout=2
               end if
            elseif(s_nex_nnew==3)then
               if(s_nex_local==0)then
                  s_nex_xout=s_nex_xin-1
                  s_nex_yout=s_nex_yin+1
                  s_nex_nout=3
                  if(s_nex_xout<0)then
                     if(s_nex_yout>side)then
                        s_nex_yout=0
                        s_nex_xout=modulo(s_nex_xout+shift,side+1)
                     else
                        s_nex_xout=side
                     end if
                  else
                     if(s_nex_yout>side)then
                        s_nex_yout=0
                        s_nex_xout=modulo(s_nex_xout+shift,side+1)
                     end if
                  end if
               elseif(s_nex_local==1)then
                  s_nex_xout=s_nex_xin
                  s_nex_yout=s_nex_yin
                  s_nex_nout=3
```

```
            end if
         end if
      case(2)
         if(s_nex_nnew==1)then
            if(s_nex_local==1)then
               s_nex_xout=s_nex_xin
               s_nex_yout=s_nex_yin
               s_nex_nout=1
            elseif(s_nex_local==0)then
               s_nex_xout=s_nex_xin
               s_nex_yout=s_nex_yin-1
               s_nex_nout=1
               if(s_nex_yout<0)then
                  s_nex_yout=side
                  s_nex_xout=modulo(s_nex_xout-shift,side+1)
               end if
            end if
         elseif(s_nex_nnew==3)then
            if(s_nex_local==1)then
               s_nex_xout=s_nex_xin
               s_nex_yout=s_nex_yin
               s_nex_nout=3
            elseif(s_nex_local==0)then
               s_nex_xout=s_nex_xin-1
               s_nex_yout=s_nex_yin
               s_nex_nout=3
               if(s_nex_xout<0)then
                  s_nex_xout=side
               end if
            end if
         end if
      case(3)
         if(s_nex_nnew==1)then
            if(s_nex_local==1)then
               s_nex_xout=s_nex_xin
               s_nex_yout=s_nex_yin
               s_nex_nout=1
            elseif(s_nex_local==0)then
               s_nex_xout=s_nex_xin+1
               s_nex_yout=s_nex_yin-1
               s_nex_nout=1
               if(s_nex_xout>side)then
                  if(s_nex_yout<0)then
                     s_nex_yout=side
                     s_nex_xout=modulo(s_nex_xout-shift,side+1)
                  else
                     s_nex_xout=0
                  end if
               else
                  if(s_nex_yout<0)then
                     s_nex_yout=side
                     s_nex_xout=modulo(s_nex_xout-shift,side+1)
                  end if
               end if
            end if
         elseif(s_nex_nnew==2)then
            if(s_nex_local==1)then
               s_nex_xout=s_nex_xin
               s_nex_yout=s_nex_yin
               s_nex_nout=2
            elseif(s_nex_local==0)then
               s_nex_xout=s_nex_xin+1
               s_nex_yout=s_nex_yin
               s_nex_nout=2
               if(s_nex_xout>side)then
```

```
                    s_nex_xout=0
                end if
            end if
        end if
    end select

    end subroutine
```

```
      subroutine output

      use defs

      implicit none
      !this writes out the properties measured in the simulation each
     time just before the temp/field changes

      integer::s_out_counter1,s_out_max
      !average all the accumulated properties
      magxtot=magxtot/(n*measurements)
      magytot=magytot/(n*measurements)
      magztot=magztot/(n*measurements)
      magtot=magtot/measurements
      magsq=magsq/measurements
      enertot=enertot/measurements
      enersq=enersq/measurements
      accept=accept/(measurements*iterations*n)
      loopaccept=loopaccept/(measurements*iterations)

      !calculate the magnitude of any vector properties
      magnetisation=sqrt(magxtot**2+magytot**2) !this is inplane
      momn=sqrt(magxtot**2+magytot**2+magztot**2)*0.75_dp !this is th
    e magnitude of the total moment including the pinned 111 spins, the 0
    .75 is because when dividing by n the n is only the spins on the plan
    e, hence 1/4 of the spins are not counted
      momnang=atan(magnetisation/magztot)*180._dp/pi !this is the ang
    le of the total moment away from the 111 axis

      !calculate secondary results
      spheat=(enersq-(enertot**2))*(1./(n*(temperature**2)))
      enertot=enertot/n
      sus=(magsq-(magtot**2))*(1./(n*temperature))
      magtot=magtot/n
      mang=fieldangle(magxtot,magytot)

      !calculate theoretical values at this temperature/field
      call theory

      if(varyt_switch==1)then !write the data as a function of temper
    ature
         write(12,'(24ES26.12E3)')temperature,enertot,magtot,accept,loo
    paccept,spheat,sus,magxtot,&
         magytot,th_mag,magnetisation,temperature/sqrt(field(1)**2+fi
    eld(2)**2+field(3)**2),&
         temperature/tkast,(temperature-tkast)**(-0.5_dp),th_sus,th_m
    agx,th_magy,th_e,th_sph,&
         mang,th_mang,magztot,momn,momnang
         write(16,'(A24,F5.3,A12,F5.3)')'Current temperature is ',temperature,' he
    aded for ',temp_stop
      elseif(varyf_switch==1)then !write the data as a function of fi
    eld strength
         write(13,'(23ES26.12E3)')f_strength,enertot,magtot,accept,loop
    accept,spheat,sus,magxtot,&
         magytot,th_mag,magnetisation,temperature/sqrt(field(1)**2+fi
    eld(2)**2+field(3)**2),&
         temperature/tkast,(temperature-tkast)**(-0.5_dp),th_sus,th_m
    agx,th_magy,th_e/3._dp,th_sph/3._dp,&
         mang,th_mang,momn,momnang
         write(16,'(A22,F6.3,A11,F6.3)')'Current field mag. is ',f_strength,'heade
    d for ',field_stop
      end if

      !write the probabilities for the loop moves
      !write(20,'(11ES26.12E3)')temperature/tkast,w0,wL,wR,eps0,eps1,
```

```
pL,pR,pO,pLR,pLL

      !write a file with the number of loops that had a particular le
ngth (cumulative frequency) at this temp or field
      s_out_max=0
      do s_out_counter1=6,n
         if(looplength_freq(s_out_counter1)/=0)then
            s_out_max=s_out_counter1
         end if
      end do
      do s_out_counter1=6,s_out_max
         if(varyt_switch==1)then
            write(19,'(ES26.15E3,2I10)')temperature,s_out_counter1,loopl
ength_freq(s_out_counter1)
         elseif(varyf_switch==1)then
            write(19,'(ES26.15E3,2I10)')f_strength,s_out_counter1,loople
ngth_freq(s_out_counter1)
         end if
      end do
      write(19,*)
      write(19,*)

      !reset all properties for the next measurement
      enertot=0._dp
      magnetisation=0._dp
      accept=0._dp
      stag_mag=0._dp
      loopaccept=0._dp
      enersq=0._dp
      magsq=0._dp
      v_magnetisation=0._dp
      v_magsq=0._dp
      looplength_freq=0
      magx=0.
      magy=0.
      magxtot=0.
      magytot=0.
      magztot=0.
      magtot=0.
      momn=0.
      momnang=0.

      call flush

      end subroutine output
```

```
      subroutine showlat

      use defs

      implicit none
      !this writes a file that can be processed with the vector plot command of gnuplot to visualise the lattice
           !it produces 4 columns, x, y, delta x, delta y to describe each spin/arrow in the lattice

      integer::s_sho_counter1,s_sho_counter2
      character(3)::s_sho_label
      real(kind=dp),dimension(3)::s_sho_r

      counter7=counter7+1
      if(counter7<10)then
         write(s_sho_label,'(I1,I1,I1)')0,0,counter7
      elseif(counter7>9.and.counter7<100)then
         write(s_sho_label,'(I1,I2)')0,counter7
      else
         write(s_sho_label,'(I3)')counter7
      end if

      open(21, file='lattice'//s_sho_label//'.out', status='replace')
      do s_sho_counter1=0,side
         do s_sho_counter2=0,side
            s_sho_r=s_sho_counter1*a1+(s_sho_counter2+0.5_dp)*a2
            if(lattice(s_sho_counter1,s_sho_counter2,1)==1)then
               write(21,'(4E20.3)')s_sho_r(1),s_sho_r(2)+0.25,0.0,-0.5
            elseif(lattice(s_sho_counter1,s_sho_counter2,1)==-1)then
               write(21,'(4E20.3)')s_sho_r(1),s_sho_r(2)-0.25,0.0,0.5
            else
               write(21,'(4E20.3)')s_sho_r(1),s_sho_r(2),0.0,0.0
            end if
            s_sho_r=s_sho_r-(0.5_dp*a2)
            if(lattice(s_sho_counter1,s_sho_counter2,2)==1)then
               write(21,'(4E20.3)')s_sho_r(1)-0.216,s_sho_r(2)-0.125,0.433,0.25
            elseif(lattice(s_sho_counter1,s_sho_counter2,2)==-1)then
               write(21,'(4E20.3)')s_sho_r(1)+0.216,s_sho_r(2)+0.125,-0.433,-0.25
            else
               write(21,'(4E20.3)')s_sho_r(1),s_sho_r(2),0.0,0.0
            end if
            s_sho_r=s_sho_r+(0.5*a1)
            if(lattice(s_sho_counter1,s_sho_counter2,3)==1)then
               write(21,'(4E20.3)')s_sho_r(1)+0.216,s_sho_r(2)-0.125,-0.433,0.25
            elseif(lattice(s_sho_counter1,s_sho_counter2,3)==-1)then
               write(21,'(4E20.3)')s_sho_r(1)-0.216,s_sho_r(2)+0.125,0.433,-0.25
            else
               write(21,'(4E20.3)')s_sho_r(1),s_sho_r(2),0.0,0.0
            end if
         end do
      end do
      close(21)

      end subroutine
```

```
      subroutine spin(s_spi_x,s_spi_y,s_spi_z,s_spi_spin,s_spi_zswitch)

      use defs, only: lattice,S1,S2,S3,Sz,dp,latt_start

      implicit none
      !this subroutine returns a vector, s_spi_spin, with the x y and z components of the spin referenced by its x and y lattice position
          !and type 1, 2 or 3. The spin is resolved into components parallel to the kagome plane and perpendicular to it, if the z switch
          !is activated it returns the perpendicular components. S1 etc are defined in the defs module.

          !usage would be call spin(x,y,z,return_vec,1/0) then use the vector when measuring magnetisation etc.

      integer, intent(in) :: s_spi_x,s_spi_y,s_spi_z,s_spi_zswitch
      real(kind=dp), dimension(3), intent(out) :: s_spi_spin

      if(s_spi_zswitch==1)then
         s_spi_spin=lattice(s_spi_x,s_spi_y,s_spi_z)*Sz
      elseif(s_spi_zswitch==0)then
         if(latt_start==6)then!set up completely ferromagnetic lattice, all spins are type 1. further info in latt_init.f90
            s_spi_spin=lattice(s_spi_x,s_spi_y,s_spi_z)*S1
         else
            select case(s_spi_z)
               case(1)
                  s_spi_spin=lattice(s_spi_x,s_spi_y,s_spi_z)*S1
               case(2)
                  s_spi_spin=lattice(s_spi_x,s_spi_y,s_spi_z)*S2
               case(3)
                  s_spi_spin=lattice(s_spi_x,s_spi_y,s_spi_z)*S3
            end select
         end if
      end if

      end subroutine spin
```

```
      subroutine theory

      use defs

      implicit none

      !this is based on PCWH's note on the K trans.
      !define the energy to flip a spin from in to out on the three t
riangle sites.
      th_mu1=2._dp*sperp*field(2)
      th_mu2=-2._dp*sperp*f_strength*cos((pi/3._dp)-fieldang)
      th_mu3=-2._dp*sperp*f_strength*cos((pi/3._dp)+fieldang)
      !define the fugacity, z_i=exp(beta*th_mu_i)
      th_z1=exp(th_mu1/temperature)
      th_z2=exp(th_mu2/temperature)
      th_z3=exp(th_mu3/temperature)
      !define alphas in terms of the fugacities, and their temperatur
e derivatives
          !the theoretical susceptibility is only valid for phi=0, be
tter to leave it out
      th_alp2=(1._dp/pi)*acos((th_z3**2-th_z2**2+th_z1**2)/(2._dp*th_
z1*th_z3))
      th_alp3=(1._dp/pi)*acos((th_z2**2-th_z3**2+th_z1**2)/(2._dp*th_
z1*th_z2))
!       th_da2dt=(-1._dp/(pi*sin(th_alp2)*temperature**2))*(((th_z3**
2-th_z1**2)*(th_mu1-th_mu3)+th_z2**2*&
!       (2._dp*th_mu2-th_mu1-th_mu3))/(th_z1*th_z3))
!       th_da3dt=(-1._dp/(pi*sin(th_alp3)*temperature**2))*(((th_z2**
2-th_z1**2)*(th_mu1-th_mu2)+th_z3**2*&
!       (2._dp*th_mu3-th_mu1-th_mu2))/(th_z1*th_z2))
      !define the theoretical quantities, need separate defs above/be
low Tk
      if(temperature<tkast)then
         th_magx=0._dp
         th_magy=2._dp*sperp/3._dp
         th_e=-(1._dp/3._dp)*th_mu1
!          th_sus=0._dp
        ! th_spheat=0._dp
      else
         th_magx=(2._dp/3._dp)*sqrt(2._dp/3._dp)*(th_alp3-th_alp2)
         th_magy=(2._dp/3._dp)*sperp*(1._dp-1.5_dp*(th_alp2+th_alp3))
         th_e=-(1._dp/3._dp)*(th_mu1+th_alp2*(th_mu2-th_mu1)+th_alp3*
(th_mu3-th_mu1))
!          th_sus=0.5*(th_magx**2+th_magy**2)**(-0.5_dp)*((((4._dp*r2
*th_magx)/(3._dp*r3))*(th_da3dt-th_da2dt))+&
!          (((-4._dp*r2*th_magy)/3._dp)*(th_da2dt-th_da3dt)))
      end if
      th_mag=sqrt(th_magx**2+th_magy**2)
      th_mang=fieldangle(th_magx,th_magy)
      th_e=th_e-0.666666_dp!this takes account of the constant intern
al energy of each tetrahedron

      end subroutine theory
```

```
## Makefile for the Spin Ice slab program
## Adam Harman-Clarke 2010

## define the macros

FORT = gfortran
COMPFLAGS = -c -Wall -fbounds-check -O
LINKFLAGS = -g -Wall -fbounds-check -o spinice
OBJS = defs.o spin.o initialise.o init_random_seed.o neighbours.o mo
ntecarlo.o loopmc.o check.o loopmove.o loopflip.o equilibrium.o equil
ibrate.o measure.o output.o finalise.o showlat.o main.o avlat.o next.
o defectcheck.o bweight.o latt_init.o theory.o

## define the suffix rules

.SUFFIXES: .o .f90
.f90.o :
        $(FORT) $(COMPFLAGS) $<
##ie to make .f90 into .o apply this rule.

%.o : %.mod
##warn make that .o depends on .mod otherwise make fails when recompi
ling the module

## define the builds

all : spinice

spinice : $(OBJS)
        $(FORT) $(LINKFLAGS) $(OBJS)

## define the object dependencies

defs.o : defs.f90
latt_init.o :defs.o latt_init.f90
bweight.o : defs.o bweight.f90
defectcheck.o : defs.o defectcheck.f90
spin.o : defs.o spin.f90
init_random_seed.o : init_random_seed.f90
neighbours.o : defs.o neighbours.f90
montecarlo.o : defs.o montecarlo.f90
loopmc.o : defs.o loopmc.f90
check.o : defs.o check.f90
loopmove.o : defs.o loopmove.f90
loopflip.o : defs.o loopflip.f90
equilibrium.o : defs.o equilibrium.f90
equilibrate.o : defs.o equilibrate.f90
initialise.o : defs.o init_random_seed.o equilibrate.o initialise.f90
measure.o : defs.o measure.f90
output.o : defs.o output.f90
finalise.o : defs.o finalise.f90
showlat.o : defs.o showlat.f90
avlat.o : defs.o avlat.f90
next.o : defs.o next.f90
theory.o : defs.o theory.f90
main.o : defs.o initialise.o init_random_seed.o equilibrate.o main.f9
0

## other

clean :
        rm defs.mod *.o spinice
```

C Neutron Scattering Code

```
      subroutine loadnew(loa_end)

      use shared

      implicit none
          !load the next spin configuration file to process

      integer,intent(out)::loa_end

      filen=filen+1
      if(filen<10)then
         write(label,'(I1,I1,I1)')0,0,filen
      elseif(filen>9.and.filen<100)then
         write(label,'(I1,I2)')0,filen
      else
         write(label,'(I3)')filen
      end if

      !read the spin sign from the input file and generate the vector
     spin orientations
      open(21,file='spin_config'//label//'data.out',status='old',err=992,io
    stat=ios)
      do ca=1,spins
         read(21,'(4I6)')cb,cc,cd,cf
         if(mod(ca-1,3)==0)then!spin 1
            lattice(ca)%orientation%component=cf*s1
         elseif(mod(ca-2,3)==0)then!spin 2
            lattice(ca)%orientation%component=cf*s2
         elseif(mod(ca,3)==0)then!spin 3
            lattice(ca)%orientation%component=cf*s3
         else
            write(22,*)'ERROR - spin not matched'
            stop
         end if
      end do
      close(21)
      loa_end=0
  992 if(ios/=0)then
         loa_end=1
      end if

      end subroutine
```

```
      program neut_scat

      use shared

      implicit none

      call setup

      do while (nomore==0)
         call loadnew(nomore)
         if(nomore==0)then
            call scatter
         end if
      end do

      call writeout

      end program

! This program takes n spin_config*n.data files that are output by th
e sislab program and processes them into a file called
! sqdata.data which can then be plotted as neutron scattering maps or
 linescans.
! It will run in two modes, either a two-dimensional map or a linesca
n as the linescan can be calculated with 10x as many points
! because it is faster.
```

```
      subroutine scatter

      use shared

      implicit none
          !this is where the calculation happens

      integer::sub

      write(22,*)'Performing a scattering measurement on spin config file',filen
      do ca=-xdiv,xdiv,xstep!loop over q space vectors
         do cb=-ydiv,ydiv
            if(lineswitch==0)then!switch for map or linescan
               sub=ca
            elseif(lineswitch==1)then
               sub=1
            end if
            if(sqrt(q(sub,cb)%component(1)**2+q(sub,cb)%component(2)**2)>qlimit.or.sqrt(q(sub,cb)%component(1)**2&
           +q(sub,cb)%component(2)**2)<qmin)then !if the q vector is larger than the max radius or smaller than the min radius of the circle/ring we want to map out then skip this iteration of the loop
               cycle
            end if
            do cc=1,spins!loop over spins
               spq=qhat(sub,cb)%component*dot_product(lattice(cc)%orientation%component,qhat(sub,cb)%component)!s perpendicular to q
               ph=exp(i*dot_product(q(sub,cb)%component,lattice(cc)%position%component))!the phase
               mqunpo=mqunpo+(lattice(cc)%orientation%component-spq)*ph!M of q unpolarised
               mqpla=mqpla+(lattice(cc)%orientation%component-spq)*(/1.,1.,0./)*ph!M of q in the kagome plane
                !  mqpar=mqpar+spq*ph !take out the parallel component as it is not physically observable and generates a 25% speed increase
               mqpseu=mqpseu+lattice(cc)%orientation%component(3)*ph!M of q pseudo spin (eqv to perpendicular to the plane)
            end do
            !one of the complex ampl. is automatically conjugated by the dot_product function
            !unpolarised scattering, ie all components of the spin perpendicular to q
            squnpo(sub,cb)=squnpo(sub,cb)+dot_product(mqunpo,mqunpo)
            !in plane perpendicular scattering, ie only the component along the y direction
            sqpla(sub,cb)=sqpla(sub,cb)+dot_product(mqpla,mqpla)
            !pseudo spin scattering, equivalent to the scattering from the components along z only
            sqpseu(sub,cb)=sqpseu(sub,cb)+dot_product(mqpseu,mqpseu)
            !parallel scattering, ie the component along the q vector only
      !     sqpar(sub,cb)=sqpar(sub,cb)+dot_product(mqpar,mqpar)
            mqpseu=0.!reset the M of q values
      !     mqpar=0.
            mqunpo=0.
            mqpla=0.
         end do
         if(ca==(-4*xdiv/5))then!useful to know how the simulation is progressing
            write(22,*)'1/10 complete...'
         elseif(ca==(-3*xdiv/5))then
            write(22,*)'2/10 complete...'
         elseif(ca==(-2*xdiv/5))then
            write(22,*)'3/10 complete...'
         elseif(ca==(-xdiv/5))then
```

```
51              write(22,*)'4/10 complete...'
52           elseif(ca==0)then
53              write(22,*)'Halfway!'
54           elseif(ca==(xdiv/5))then
55              write(22,*)'6/10 complete...'
56           elseif(ca==(2*xdiv/5))then
57              write(22,*)'7/10 complete...'
58           elseif(ca==(3*xdiv/5))then
59              write(22,*)'8/10 complete...'
60           elseif(ca==(4*xdiv/5))then
61              write(22,*)'9/10 complete...'
62           end if
63           call flush
64           if(lineswitch==1)then
65              exit
66           end if
67        end do
68        write(22,*)'Finished S(Q) correlations measurement'
69
70        end subroutine
```

```
      subroutine setup

      use shared

      implicit none

      open(22,file='progress.txt')

      !find out how many spins are in the lattice
      open(20,file='spin_config001data.out')
      do ca=1,50000
         read(20,'(4I6)',end=991)cb,cc,cd,ce
         spins=spins+1
      end do
      write(22,*)'ERROR - More than 50000 spins in the lattice!'
      stop
991   write(22,*)'Number of spins in the lattice',spins
      close(20)
      side=(nint(sqrt(real(spins)/3.0))-1)

      !allocate the lattice
      allocate(lattice(1:spins))

      !generate the positions of the spins using the lattice vectors
      cd=0
      do ca=0,side
         do cb=0,side
            do cc=1,3
               cd=cd+1
               select case(cc)
                  case(1)
                     lattice(cd)%position%component=(real(ca)*a1)+((real(cb)+0.5)*a2)
                  case(2)
                     lattice(cd)%position%component=(real(ca)*a1)+(real(cb)*a2)
                  case(3)
                     lattice(cd)%position%component=((real(ca)+0.5)*a1)+(real(cb)*a2)
               end select
            end do
         end do
      end do

      !determine how big to make the scattering map in q space and how many divisions to make along the edges of the scattering map
      read(*,*)lineswitch
      if(lineswitch==0)then ! generating a map in q space...
         read(*,*)xqmult !if the multipliers = 2 then the map is 4pi*b1 x 4pi*b2 ~~ -2pi to 2pi on each side
         read(*,*)yqmult
         xdiv=nint(2.*xqmult*real((side+1))) !these control how fine the q mesh is
         ydiv=nint(2.*yqmult*real((side+1)))

         !allocate the scattering intensity matrices and the q vector matrices
         allocate(squnpo(-xdiv:xdiv,-ydiv:ydiv),sqpla(-xdiv:xdiv,-ydiv:ydiv),sqpar(-xdiv:xdiv,-ydiv:ydiv),&
         sqpseu(-xdiv:xdiv,-ydiv:ydiv),q(-xdiv:xdiv,-ydiv:ydiv),qhat(-xdiv:xdiv,-ydiv:ydiv))

         !generate the q vectors
         do ca=-xdiv,xdiv
            do cb=-ydiv,ydiv
```

```
                q(ca,cb)%component=((real(ca)/real(xdiv))*xqmult*b1)+(
(real(cb)/real(ydiv))*yqmult*b2)
                if((q(ca,cb)%component(1)**2+q(ca,cb)%component(2)**2)
<0.00001)then
                   qhat(ca,cb)%component=0.
                else
                   qhat(ca,cb)%component=q(ca,cb)%component/sqrt(q(ca,
cb)%component(1)**2+q(ca,cb)%component(2)**2)
                end if
             end do
          end do
          qlimit=abs(q(-xdiv,0)%component(1))!these are the inner and
outer limits determining the ring of q space
          read(*,*)qmin
          qmin=qmin*2.0*pi
       elseif(lineswitch==1)then ! ...else generating a linescan throu
gh q space.
          read(*,*)xqmult !if the multipliers = 2 then the map is 4pi*
b1 x 4pi*b2 ~~ -2pi to 2pi on each side
          read(*,*)yqmult !so the line will have this extent as well.
          read(*,*)qmin !this is not applicable for lines so set it to
 0
          qmin=0.
          read(*,*)hswitch !if hswitch =1 then the line is at constant
 -h,h,0 otherwise it is at constant -k,-k,2k
          read(*,*)h_or_k !this is the value in the display coords
          ydiv=10*nint(2.*yqmult*real(side+1)) !this is a general coun
ter now and does not refer to the y direction! both lines use this
          xdiv=1                                !it still controls how
fine the q mesh/line is, yqmult controls how far it extends
          xstep=2
          !allocate the scattering intensity matrices and the q vector
 matrices
          allocate(squnpo(1,-ydiv:ydiv),sqpla(1,-ydiv:ydiv),sqpar(1,-y
div:ydiv),&
          sqpseu(1,-ydiv:ydiv),q(1,-ydiv:ydiv),qhat(1,-ydiv:ydiv))
          !generate the q vectors
          do ca=-ydiv,ydiv
             if(hswitch==1)then !vertical linescan
                q(1,ca)%component=(h_or_k*b1)+((real(ca)/real(ydiv))*y
qmult*b2) !fixed x, then all values of y
             elseif(hswitch==0)then !horizontal linescan
                q(1,ca)%component=((real(ca)/real(ydiv))*yqmult*2.*pi*
(/1.,0.,0./))+(h_or_k*(3./2.)*b2) !all values of a hori line, then fi
xed y
             end if
             if((q(1,ca)%component(1)**2+q(1,ca)%component(2)**2)<0.00
001)then
                qhat(1,ca)%component=0.
             else
                qhat(1,ca)%component=q(1,ca)%component/sqrt(q(1,ca)%co
mponent(1)**2+q(1,ca)%component(2)**2)
             end if
          end do
          qlimit=99999.
       end if
       end subroutine setup
```

```
1       module shared
2
3       implicit none
4
5       integer::spins=0
                    !the number of spins in the lattice
6       integer::xdiv=0,ydiv=0
                    !number of divisions of each edge of s(q) map
7       integer::xstep=1
                    !step for loops, prevents calculating twice with
    lines
8       integer::ca=0,cb=0,cc=0,cd=0,ce=0,cf=0
                    !counters for loops, general use etc
9       integer::lineswitch,hswitch
                    !switches for line or plane of q space and which
    line
10      real::h_or_k
                    !value of constant for line scan, depends on hswi
    tch
11      type vector
                    !define a vector object
12         real,dimension(3)::component
                    !it has three components
13      end type vector
14      type spin
                    !define a spin object
15         type(vector)::orientation
                    !vector describing the spin orientation
16         type(vector)::position
                    !vector describing its lattice position
17      end type spin
18      type(spin),dimension(:),allocatable::lattice
                    !the lattice of spins
19      real,parameter::r2=sqrt(2.),r3=sqrt(3.),r6=sqrt(6.)
                    !useful square roots
20      real,parameter::pi=acos(-1.)
                    !pi
21      complex,parameter::i=(0.,1.)
                    !i, sqrt(-1)
22      real,dimension(3),parameter::a1=(/1.,0.,0./),a2=(/0.5,r3/2.,0./
    )               !basis vectors for the lattice
23      real,dimension(3),parameter::b1=2.*pi*(/1.,-1./r3,0./),b2=2.*pi
    *(/0.,2./r3,0./)    !reciprocal lattice vectors
24      real,dimension(3),parameter::s1=(2.*r2/3.)*(/0.,-1.,1./(2.*r2)/
    )               !vectors describing the spins
25      real,dimension(3),parameter::s2=(2.*r2/3.)*(/r3/2.,0.5,1./(2.*r
    2)/)            !when positive they point into the center of the
26      real,dimension(3),parameter::s3=(2.*r2/3.)*(/-r3/2.,0.5,1./(2.*
    r2)/)           !tetrahedron
27      real::xqmult=0.,yqmult=0.
                    !multipliers for the q space vectors
28      integer::side=0
                    !there are 0 to side triangles along one lattice
    edge
29      type(vector),dimension(:,:),allocatable::q,qhat
                    !the q vector and q hat
30      real::qlimit,qmin
                    !the max and min radius of the circular q map
31      real,dimension(3)::spq
                    !the component of S along q
32      complex::ph
                    !the complex scattering phase factor
33      complex,dimension(3)::mqunpo,mqpar,mqpla,mqpseu
                    !scat amp, unpol, inplane unpo, pseudospin, paral
    lel
```

```
34        real,dimension(:,:),allocatable::squnpo,sqpar,sqpla,sqpseu
                      !scat int, as above
35        integer::filen=0
                      !counter for the spin config files
36        character(3)::label
                      !label for the above counter
37        integer::nomore=0
                      !switch for reaching the last spin config file
38        integer::ios=0
                      !error number for file opening
39
40        end module
```

```
      subroutine writeout

      use shared

      implicit none
      !this writes the calculated values to a file
      integer::sub

      squnpo=squnpo/(filen*spins)!normalise by the number of snapshot
s and the number of spins in the lattice to get per spin values
      sqpla=sqpla/(filen*spins)
      sqpar=sqpar/(filen*spins)
      sqpseu=sqpseu/(filen*spins)

      open(23,file='cir_sqdata.out',status='new')
      do ca=-xdiv,xdiv!loop over the reciprocal space positions
         do cb=-ydiv,ydiv
            if(lineswitch==0)then!if it is a map or a linescan...
               sub=ca
            elseif(lineswitch==1)then
               sub=1
            end if
            if(sqrt(q(sub,cb)%component(1)**2+q(sub,cb)%component(2)*
*2)>qlimit.or.sqrt(q(sub,cb)%component(1)**2+&
            q(sub,cb)%component(2)**2)<qmin)then!write # for values o
utside the required ring
               write(23,'(2ES26.15E3,3A26)')q(sub,cb)%component(1),q(sub
,cb)%component(2),'#','#','#'
            else
               write(23,'(5ES26.15E3)')q(sub,cb)%component(1),q(sub,cb)
%component(2),squnpo(sub,cb),sqpla(sub,cb),sqpseu(sub,cb)
            end if
         end do
         write(23,*)
         if(lineswitch==1)then
            exit
         end if
      end do

      end subroutine
```

```
#makefile for the neutron scattering program
neut : main.o shared.o setup.o scatter.o loadnew.o writeout.o
        gfortran -o neut *.o

main.o : shared.o main.f90
        gfortran -c main.f90
loadnew.o : shared.o loadnew.f90
        gfortran -c loadnew.f90
writeout.o : shared.o writeout.f90
        gfortran -c writeout.f90
shared.o : shared.f90
        gfortran -c shared.f90
setup.o : shared.o setup.f90
        gfortran -c setup.f90
scatter.o : shared.o scatter.f90
        gfortran -c scatter.f90

clean :
        rm *.o neut *.mod
```

D Square Ice Code

```
/*
 *  Globals.h
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 23/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#ifndef GLOBALS_H
#define GLOBALS_H

#include<vector>

//Variables

extern int mcs;//number of monte carlo steps
extern double run_ener;//running total of the energy
extern double temperature;
extern vector<double> field;
extern vector<double> satfield;//saturation field

extern double exchj; //the exchange interaction energy term, it varies with temp, at t=0 it is exchint (below)
extern double e_fourzero;  //the energy of a four in or out vertex.
extern double e_threeone;  //the energy of a three in / three out vertex
extern double e_4twotwo;   //the energy of a two in / two out vertex in the 4 vertex model
extern double e_6twotwo;   //the energy of a two in / two out vertex in the 6 vertex model.
/*calculating the energies of the vertices from the exchange model gives
 *E(four in, four out)=-2.*exchint
 *E(three in/out, one out/in)=0.
 *E(two in/out, two in/out, net moment)=2.*exchint
 *E(two in/out, two in/out, no moment)=-2.*exchint
 *ie the two 2in/2out vertices with no net moment (spins antiparallel) are the same energy as the 4in/0out vertices. to get the 6 vertex model set them equal to the other 2in/2out vertices.
 *
 *calculating the energies of the vertices from the dipolar interaction means they change with temperature. the difference is that the 2in/2out no moment vertices are now the groundstate of the model and in a field they are not much above the other 2in/2out vertices, which become the lowest energy
 */


//Constants

const double pi=4.*atan(1.);  //pi
const double kb=1.38e-23;    //boltzmann's constant
const double mub=9.274e-24;  //bohr magneton
const double mu=1.2e7*mub;   //the moment of an island (maximum)
const double l=750e-9;       //the length of an island in meters
const double nnd=300e-9;     //nearest neighbour distance between center of mass of the island at the end for monopole interactions
const double nnnd=424.26e-9;  //next nearest neighbour distance, nnd is between perp islands, nnnd is between parallel islands
const double coul_fact=(10e-7/kb)*pow((mu/l), 2.); //the constant factor in the coulomb energy calculation. E_coul is then +-coul_fact*splength^2/distance
const double tcrit=230.;    //critical temperature of the material
const double beta=0.3333;    //critical exponent for spin length func
```

```
tion
const double exchint=207.;   //exchange interaction between islands in K
const double dipmom=1.;//-(coul_fact/4.)*((2.-sqrt(2.))/nnd);      for simplicity currently set to 1
//this should be equal to mu but is set to absorb the factors from the monopole energy such that a field of (1,1) is enough to make the 1,1 ordered state the GS at all temps

#endif
```

```
/*
 *  Globals.cpp
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 23/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#include<vector>

using namespace std;

int mcs;
double run_ener;
double temperature;
vector <double> field;
vector <double> satfield;
double exchj;
double e_fourzero;
double e_threeone;
double e_4twotwo;
double e_6twotwo;
```

```
/*
 *  Lattice.h
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 08/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#ifndef LATTICE_H
#define LATTICE_H

#include <vector>
#include "Spin.h"
#include "Globals.h"

class Lattice{
private:
        int m_n; //number of spins in the lattice
        int side; //the number of lattice sites along one edge of the
 lattice
        vector<Spin> slist; // vector of all the spins in the lattice
        void Neighbours (); //work out the neighbours of each spin an
d store them
        int *nbrs;
        vector<double> vals; //vector to store the measurement data
        double ener;
        double magn;
        double magx;
        double magy;
        double chxmag; // variables for chain magnetisation
        double chymag;
        int outcalled;
        static int lattcount;
        string intact; //variable to control the interaction type
        double splength;
public:
        Lattice();
        ~Lattice();
        int Size () {return m_n;}
        void Mcupdate();
        void Magnetisation(double&, double&, double&, double&, double
&);
        void Output();
        void Measure();
        double ExchEnergy(Spin);
        double VertEnergy(Spin);
        void Moment();
        void Lattfile();
        void param_update();
        void saturate(int&);
};

#endif
```

```
/*
 *  Lattice.cpp
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 08/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#include <cmath>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <sstream>
#include <string>
#include "Lattice.h"
#include "Spin.h"
#include "stblib.h"
#include "Utilities.h"
#include "Globals.h"

int Lattice::lattcount=0;

//constructor
Lattice::Lattice(){
        nbrs=NULL;
        m_n=read_single_input_string<int>("number of spins in the lattice");
        slist.push_back(Spin(0.,0.,0.,0.));//slist is a one D vector of all the spins in the lattice
        string conf = read_single_input_string<string>(string("starting configuration"));
        string intact = read_single_input_string<string>(string("interaction type"));
        side=int(sqrt(m_n/2.));//the number of islands along an edge, ie in a lattice of 18 spins this is 3
        outcalled=0;
        ener=0.0;
        magn=0.0;
        magx=0.0;
        magy=0.0;
        chxmag=0.0;
        chymag=0.0;
        for (int i=1; i<=m_n; i++) {//determine the position and spin hori and vert components for each spin
                double py=(i-1)/(2*side);
                double px=((i-1)/2)-py*side; // the position is the point where the spins meet
                double vsx=0.0;
                double vsy=1.0;
                double hsx=1.0;
                double hsy=0.0; // these default options are the ordered11 option
                if (conf=="random") {
                        double rand=stb_rand();
                        if (rand<=0.5) vsy=-1.0;
                        rand=stb_rand();
                        if (rand<=0.5) hsx=-1.0;
                }
                if (conf=="randomvertical") {
                        double rand=stb_rand();
                        if (rand<=0.5) vsy=-1.0;
                }
                if (conf=="dipolegs") {
                        double row=ceil(double(i)/(2.*double(side)));
                        double col=ceil(double((i-((row-1)*(2*side)))
```

```
/2.));
                        vsy=pow(-1.,(row-1.))*pow(-1.,(col-1.));
                        hsx=pow(-1.,row)*pow(-1.,(col-1.));
                }
                if (i%2!=0){ //vertical spin
                        slist.push_back(Spin(px,py,vsx,vsy));
//                      cout<<"Created a vert spin with components: "
<<px<<" "<<py+0.5<<" "<<vsx<<" "<<vsy<<endl;
                }
                else { //horizontal spin
                        slist.push_back(Spin(px,py,hsx,hsy));
//                      cout<<"Created a hori spin with components: "
<<px+0.5<<" "<<py<<" "<<hsx<<" "<<hsy<<endl;
                }
        }
        Neighbours();//work out the neighbours of each spin
        param_update();
        //calculate the energy of the lattice
        run_ener=0.0;
        for (int i=1; i<=m_n; i++) {
                if (intact=="exchange") {
                        run_ener+=ExchEnergy(slist[i]);
                }
                else {
                        run_ener+=VertEnergy(slist[i]);
                }
        }
}

//destructor
Lattice::~Lattice(){
        delete[] nbrs;
        nbrs=NULL;
}

/*
 *--  Member functions
 */

//Private: work out the id of the neighbours of each spin and store i
t
void Lattice::Neighbours(){
        try {
                nbrs = new int[(6*m_n)+5];
        }
        catch (bad_alloc&) {
                cout << "nbrs could not be allocated.";
                cout.flush();
                exit(1);
        }
        int row = 2*side; //the number of spins along a row of the la
ttice, ie in a lattice of 18 this is 6
        //the array nbrs is an int list of the reference to the spins
 in the slist which are the six neighbours of each spin.
        //the array is 1D, format is s_id*6, s_id*6+1 ... s_id*6+5 fo
r the 6 neighbours of spin with s_id.
        //6-12 are neighbours of the first spin etc. the first 3 neig
hbours are one vertex, 2nd 3 the other.
        for (int i=0;i<=m_n;i++){ //loop over the lattice
                if(i%2!=0) { //vert spin, neighbours of spin n are, n
+1, n-r, n-1, n+r-1, n+r, n+r+1
                        nbrs[(i*6)]=i+1;
                        nbrs[(i*6)+1]=i-row;
                        if(i-row<1) nbrs[(i*6)+1]+=m_n;
                        nbrs[(i*6)+2]=i-1;
```

```
                        if((i-1)%row==0) nbrs[(i*6)+2]+=row;
                        nbrs[(i*6)+3]=(i+row)-1;
                        if((i+row)-1>m_n) nbrs[(i*6)+3]-=m_n;
                        if(((i+row)-1)%row==0) nbrs[(i*6)+3]+=row;
                        if(i==(m_n-(row-1))) nbrs[(i*6)+3]=row;
                        nbrs[(i*6)+4]=i+row;
                        if(i+row>m_n) nbrs[(i*6)+4]-=m_n;
                        nbrs[(i*6)+5]=i+row+1;
                        if(i+row+1>m_n) nbrs[(i*6)+5]-=m_n;
                }
                else { //hori spin, neighbours of spin n are, n+1, n+2, n-r+1, n-r-1, n-2, n-1
                        nbrs[(i*6)]=i+1;
                        nbrs[(i*6)+1]=i+2;
                        nbrs[(i*6)+2]=(i-row)+1;
                        if(i%row==0) {
                                nbrs[(i*6)]-=row;
                                nbrs[(i*6)+1]-=row;
                                nbrs[(i*6)+2]-=row;
                        }
                        if(i-row+1<2) nbrs[(i*6)+2]+=m_n;
                        if(i==row) nbrs[(i*6)+2]=m_n-(row-1);
                        nbrs[(i*6)+3]=(i-row)-1;
                        if((i-row)-1<0) nbrs[(i*6)+3]+=m_n;
                        nbrs[(i*6)+4]=i-2;
                        if((i-2)%row==0) nbrs[(i*6)+4]+=row;
                        nbrs[(i*6)+5]=i-1;
                }
        }
}

//Monte Carlo single spin flip update
void Lattice::Mcupdate(){
        double rand=stb_rand();
        int choosespin=equal_prob_ints(m_n);
        while ((choosespin<1) || (choosespin>m_n)) {
                choosespin=equal_prob_ints(m_n);
        }
        double eold,enew,deltae;
        if (intact=="exchange") {
                eold=ExchEnergy(slist[choosespin]);
                deltae=-2.*eold;
                if (deltae<0.0 || rand<exp(-deltae/temperature)) {
                        slist[choosespin].Flipspin();
                        run_ener+=deltae;
                }
        }
        else {
                eold=VertEnergy(slist[choosespin]);
                Spin flip=slist[choosespin];
                flip.Flipspin();
                enew=VertEnergy(flip);
                deltae=enew-eold;
                if (deltae<0.0 || rand<exp(-deltae/temperature)) {
                        slist[choosespin]=flip;
                        run_ener+=deltae;
                }
        }
}

//Calculate the energy of a spin

//this function calculates the energy of a spin based on the vertex at either end of that spin
double Lattice::VertEnergy(Spin spin){
```

```
178         //examine the spins at the vertex at each end of a spin to de
    termine what type it is, look up the energy of both the v types
179         int s_id=spin.Id();
180         vector<double>s_one=spin.Getdir();
181         vector<double> vertone=s_one;
182         vector<double> verttwo=s_one;
183         vector<double> sn_one(2,0.0),sn_two(2,0.0),sn_three(2,0.0),sn
    _four(2,0.0),sn_five(2,0.0),sn_six(2,0.0);//sn_one etc. are the 6 nei
    ghbours of spin, 1-3 are right/bottom vert, 4-6 are left/top v
184         sn_one=slist[nbrs[s_id*6]].Getdir();
185         sn_two=slist[nbrs[s_id*6+1]].Getdir();
186         sn_three=slist[nbrs[s_id*6+2]].Getdir();
187         sn_four=slist[nbrs[s_id*6+3]].Getdir();
188         sn_five=slist[nbrs[s_id*6+4]].Getdir();
189         sn_six=slist[nbrs[s_id*6+5]].Getdir();
190         //vertone/two are the sum of the spin plus its three neighbou
    rs
191         vertone=vec_add(vertone,sn_one);
192         vertone=vec_add(vertone,sn_two);
193         vertone=vec_add(vertone,sn_three);
194         verttwo=vec_add(verttwo,sn_four);
195         verttwo=vec_add(verttwo,sn_five);
196         verttwo=vec_add(verttwo,sn_six);
197         double eone,etwo; //the energy of the two vertices
198         if (sqrt(vertone[0]*vertone[0]+vertone[1]*vertone[1])<0.00000
    01){ //this is a head to head vertex or an all in/out vertex
199                 if (s_one[0]==0.0) {//then the spin under considerati
    on is vertical
200                         if (abs(sn_one[0]+sn_two[1])>0.0) {
201                                 eone=e_6twotwo;
202 //                              cout << "Vert r/b is two in two out,
    6vertex, ";
203                         }
204                         else {
205                         eone=e_fourzero;
206 //                          cout << "Vert r/b is four in / out, ";
207                     }
208                 }
209                 else {//it's horizontal
210                         if (abs(sn_one[1]+sn_two[0])>0.0) {
211                                 eone=e_fourzero;
212 //                              cout << "Vert r/b is four in / out, "
    ;
213                         }
214                         else {
215                                 eone=e_6twotwo;
216 //                              cout << "Vert r/b is two in two out,
    6vertex, ";
217                         }
218         }
219         }
220         else if (abs(vertone[0])>0.0 && abs(vertone[1])>0.0){
221                 eone=e_4twotwo;
222 //              cout << "Vert r/b is two in two out, 4vertex, ";
223         }
224         else {
225                 eone=e_threeone;
226 //              cout << "vert r/b is three in / out, ";
227         }
228         if (sqrt(verttwo[0]*verttwo[0]+verttwo[1]*verttwo[1])<0.00000
    01){ //this is a head to head vertex or an all in/out vertex
229                 if (s_one[0]==0.0) {//then the spin under considerati
    on is vertical
230                         if (abs(sn_one[0]+sn_two[1])>0.0) {
231                                 etwo=e_6twotwo;
```

```
232 //                              cout << "Vert l/t is two in two out 6
    vertex, ";
233                         }
234                         else {
235                         etwo=e_fourzero;
236 //                          cout << "Vert l/t is four in / out, ";
237                     }
238                 }
239                 else {//it's horizontal
240                         if (abs(sn_one[1]+sn_two[0])>0.0) {
241                                 etwo=e_fourzero;
242 //                              cout << "Vert l/t is four in / out, "
    ;
243                         }
244                         else {
245                                 etwo=e_6twotwo;
246 //                              cout << "Vert l/t is two in two out 6
    vertex, ";
247                         }
248         }
249         }
250         else if (abs(verttwo[0])>0.0 && abs(verttwo[1])>0.0){
251                 etwo=e_4twotwo;
252 //              cout << "Vert l/t is two in two out 4vertex, ";
253         }
254         else {
255                 etwo=e_threeone;
256 //              cout << "vert l/t is three in / out, ";
257         }
258         double venergy=-(eone+etwo)-dipmom*dot(vec_add(vertone,verttw
    o),field);//energy is the sum of the two vertices and the field inter
    action
259 //      cout << "eone is "<<eone<<" and etwo is "<<etwo<<endl;
260 //      cout << "energy of spin is: "<<venergy<<endl;
261         return venergy;
262         }
263
264 //-----this is the function for nearest neighbour spin exchange energ
    ies, addition of a cross product term could represent n.n. dipoles
265 double Lattice::ExchEnergy(Spin spin){
266         //find its neighbours
267         vector<double> neigh(2,0.0);
268         int s_id=spin.Id();
269         vector<double> s_one=spin.Getdir();
270         for (int i=0; i<6; i++) {
271                 vector<double> temp=slist[nbrs[s_id*6+i]].Getdir();
272                 neigh=vec_add(neigh,temp);
273         }
274         //calculate the energy
275         double exenergy=-exchint*dot(s_one,neigh)-dipmom*dot(s_one,fi
    eld);
276         return exenergy;
277 }
278
279 //Calculate the various magnetisations of the lattice, in the x direc
    tion, y direction, length of moment(in y dir), chain mag
280 void Lattice::Magnetisation(double& xmag, double& ymag, double& nmag,
     double& chx, double& chy){
281         vector<double> magn(2,0.0),temp(2,0.0);
282         double vchains[(side-1)];
283         double hchains[(side-1)];//position 0 represents the leftmost
    /bottom chain etc, pos side-1 is the rmost/top chain
284         for (int i=0; i<side; i++) {
285                 vchains[i]=0.0;
286                 hchains[i]=0.0;
```

```
287         }
288         int ref;
289         for (int i=1; i<=m_n; i++) {//condense the magnetisation of e
    ach hori or vert chain to one var, then sum the abs value of them to
    get an overall hori and vert chain magnitude
290                 temp=slist[i].Getdir();//the vector specifiying the c
    urrent spin's direction
291                 magn=vec_add(magn,temp);
292 //              cout << "spin number "<<i<<", "<<endl;
293                 if (i%2!=0) {//vertical spins, ref = 1,3,5,7...side-1
294                         if (i<(2*side)) {
295                                 ref=i;
296                         }
297                         else {
298                                 ref=i%(2*side);
299                         }
300 //                      cout << "ref= "<<ref<<endl;
301                         vchains[int((ref-1)/2.0)]+=temp[1];
302 //                      cout<<"vchains ref "<<int((ref-1)/2.0)<<endl;
303                 }
304                 else {//horizontal spins
305                         ref=int(floor(double(i)/(2.0*double(side))));
306                         if (i%(2*side)==0) ref-=1;
307 //                      cout << "ref= "<<ref<<endl;
308                         hchains[ref]+=temp[0];
309 //                      cout<<"hchains ref"<<ref<<endl;
310                 }
311         }
312         chx=0.0;
313         chy=0.0;
314         for (int i=0; i<side; i++) {
315                 chx+=abs(hchains[i])/(side*side);
316                 chy+=abs(vchains[i])/(side*side);
317 //              cout<<"chxmag, chymag "<<chxmag<<" "<<chymag<<endl;
318         }
319         xmag=magn[0];
320         ymag=magn[1];
321         nmag=sqrt(pow(xmag, 2.0)+pow(ymag, 2.0));
322 }
323
324 //Measure quantities of the lattice and output them
325 void Lattice::Output(){
326         if(outcalled<2) outcalled++;
327         //calculate quantities that do not need averaging over measur
    ements
328         double bmag=sqrt(pow(field[0],2)+pow(field[1],2));
329         if(field[1]<0) bmag*=-1.;
330         vector<double>spinsize=slist[1].Getdir();
331     double spsize=(spinsize[0]==0) ? abs(spinsize[1]) : abs(spinsize[
    0]);
332         //set up the file for output
333         ofstream out("resultsdata.out", ios::app);
334         if (!out) cerr<<"couldn't open the file!"<<endl;
335         if(outcalled==1){
336                 //write column labels
337                 out<<setw(20)<<"#    1.Temperature"<<",";
338                 out<<setw(20)<<"2.Field Magn."<<",";
339                 out<<setw(20)<<"3.Field x"<<",";
340                 out<<setw(20)<<"4.Field y"<<",";
341                 out<<setw(20)<<"5.Norm mag x comp."<<",";
342                 out<<setw(20)<<"6.Norm mag y comp."<<",";
343                 out<<setw(20)<<"7.Mag vector length"<<",";
344                 out<<setw(20)<<"8.Energy (exchJ)"<<",";
345                 out<<setw(20)<<"9.Spin length"<<",";
346                 out<<setw(20)<<"10.M_rem/M_sat"<<",";
```

```
347                 out<<setw(20)<<"11.Norm mag"<<",";
348                 out<<setw(20)<<"12.Norm xchainmag"<<",";
349                 out<<setw(20)<<"13.Norm ychainmag"<<",";
350                 out<<setw(20)<<"14.Norm chainmag"<<",";
351                 out<<setw(20)<<"15.1D Ising mag"<<",";
352                 out<<endl;
353         }
354         //divide by the number of measurements and m_n to make quanti
    ties extrinsic and write them to the file
355         out.setf(ios::fixed);
356         out<<setw(20)<<setprecision(10)<<temperature<<",";
357         out<<setw(20)<<setprecision(10)<<bmag<<",";
358         out<<setw(20)<<setprecision(10)<<field[0]<<",";
359         out<<setw(20)<<setprecision(10)<<field[1]<<",";
360         out<<setw(20)<<setprecision(10)<<(2.0*magx)/(spsize*m_n*mcs)<
    <",";//x2 because there are only m_n/2 x-spins
361         out<<setw(20)<<setprecision(10)<<(2.0*magy)/(spsize*m_n*mcs)<
    <",";
362         out<<setw(20)<<setprecision(10)<<(2.0*magn)/(m_n*mcs)<<",";
363         out<<setw(20)<<setprecision(10)<<ener/(exchint*m_n*mcs)<<",";
364         out<<setw(20)<<setprecision(10)<<spsize<<",";
365         if (temperature<tcrit) {
366                 out<<setw(20)<<setprecision(10)<<abs(magn)/(m_n*mcs*s
    psize)<<",";
367         }
368         else {
369                 out<<setw(20)<<0.0<<",";
370         }
371         out<<setw(20)<<setprecision(10)<<((2.0*abs(magn))/(m_n*mcs))/
    sqrt(pow((2.0*magx)/(spsize*m_n*mcs),2.0)+pow((2.0*magy)/(spsize*m_n*
    mcs),2.0))<<",";
372 //      if (field[1])>0.0) {
373 //              out<<setw(20)<<setprecision(10)<<(abs(magn)*sqrt(2.0)
    )/(m_n*mcs)<<",";
374 //      }
375 //      else {
376 //              out<<setw(20)<<setprecision(10)<<(abs(magn)*2.0)/(m_n
    *mcs)<<",";
377 //      }
378         //normalised chain magnetisation
379         out<<setw(20)<<setprecision(10)<<chxmag/(spsize*mcs)<<",";
380         out<<setw(20)<<setprecision(10)<<chymag/(spsize*mcs)<<",";
381         out<<setw(20)<<setprecision(10)<<sqrt(pow(chxmag,2.0)+pow(chy
    mag,2.0))/(sqrt(2.)*spsize*mcs)<<",";
382         out<<setw(20)<<setprecision(10)<<sqrt(exp((-side/2.0)*exp((-2
    ./temperature)*exchint*pow((1.0-(temperature/tcrit)),(2.0*beta))))+po
    w((1.0-(temperature/tcrit)),beta)/side)<<",";
383         out<<endl;
384         out.close();
385         //reset the totals for the next temp/field
386         ener=0.;
387         magx=0.;
388         magy=0.;
389         magn=0.;
390         chxmag=0.;
391         chymag=0.;
392 }
393
394 //Measure lattice quantities
395 void Lattice::Measure(){
396         //cout << "Called Lattice::Measure"<<endl;
397         ener+=run_ener;
398         double mx, my, ml, mcx, mcy;
399         Magnetisation(mx,my,ml,mcx,mcy);
400         magx+=mx;
```

```
401         magy+=my;
402         magn+=ml;
403         chxmag+=mcx;
404         chymag+=mcy;
405 }
406
407
408 //Update parameters which change everytime the temperature or field c
    hanges
409 void Lattice::param_update(){
410         splength=pow((tcrit-temperature)/tcrit,beta);
411 //      splength=1;
412
413         //The energy of a spin is -(E(vert 1) + E(vert 2)) so in the
    following definitions +ve means most favourable
414         //energies calculated from parallel exchange interactions
415     //the exchange interaction between spins reduced by the same func
    tion as the spin length to 0 at Tc
416         exchj=exchint*splength*splength;
417 //      exchj=exchint;
418         e_fourzero=-2.0*exchj;
419         e_threeone=0.0;
420         e_4twotwo=2.0*exchj;
421         e_6twotwo=-2.0*exchj;
422         //energies calculated with point dipole interactions
423         //double dip=9.9080347*splength*splength;//D=(mu0*mu^2)/(4*pi
    *d^3) which is (1.44*86.007*10E-17*splength*splength)/(125.), multipl
    y by 10E18 to get D of order 10
424 //      e_fourzero=-14.*dip*splength*splength;
425 //      e_threeone=-2.*dip*splength*splength;
426 //      e_4twotwo=2.*dip*splength*splength;
427 //      e_6twotwo=10.*dip*splength*splength;
428         //energies calculated from monopole interactions
429 //      e_fourzero=-coul_fact*splength*splength*(4./nnd + 2./nnnd);
430 //      e_threeone=0.;
431 //      e_4twotwo=-coul_fact*splength*splength*(-2./nnnd);
432 //      e_6twotwo=-coul_fact*splength*splength*(-4./nnd+2./nnnd);
433
434 //Update the moments to reflect the temperature, ie moment decreases
    until a critical temp where it vanishes
435         for (int i=1;i<=m_n;i++) {
436                 if (temperature<=tcrit) {
437                         vector<double>momn=slist[i].Getdir();
438                         //                      cout << "moment in "<
    <momn[0]<<momn[1]<<endl;
439                         if (momn[0]!=0.) momn[0]*=splength/abs(momn[0
    ]);
440                         if (momn[1]!=0.) momn[1]*=splength/abs(momn[1
    ]);
441                         //                      cout << "moment out "
    <<momn[0]<<momn[1]<<endl;
442                         slist[i].Setdir(momn[0],momn[1]);
443                 }
444                 else {
445                         slist[i].Setdir(0.,0.);
446                 }
447     }
448 }
449
450 //Write a file that can be plotted to show the state of the lattice
451 void Lattice::Lattfile(){
452         //the format of the file is x, y, dx, dy, then gnuplot 'with
    vectors'
453         int filenum=lattcount++;
454         stringstream temp;
```

```
455         temp<<filenum;
456         string label="Lattice";
457         label+=temp.str()+".out";
458         ofstream out (label.c_str());
459         for (int i=1; i<=m_n; i++) {
460                 vector<double>pos=slist[i].Getpos();
461                 //the position of all spins is the point where they m
    eet, ie the bravais lattice point, ie the bottom left corner of a squ
    are
462                 vector<double>dir=slist[i].Getdir();
463                 double s_len=(dir[0]==0.0) ? abs(dir[1]) : abs(dir[0]
    );
464                 //the spin direction then indicates the spin vector a
    t that point. we need to adjust the starting position so the spins ar
    e always centered halfway between lattice points
465                 if (i%2!=0.0) { //it's a vertical spin
466                         out<<setw(20)<<setprecision(10)<<pos[0]<<" ";
467                         if (dir[1]<0.0) {//pointing down
468                                 out<<setw(20)<<setprecision(10)<<pos[
    1]+0.5+0.5*s_len;
469                         }
470                         else {//pointing up
471                                 out<<setw(20)<<setprecision(10)<<pos[
    1]+0.5-0.5*s_len;
472                         }
473                         out<<setw(20)<<setprecision(10)<<0.0<<" ";
474                         out<<setw(20)<<setprecision(10)<<dir[1]<<endl
    ;
475                 }
476                 else { //it's a horizontal spin
477                         if (dir[0]<0.0) {//pointing left
478                                 out<<setw(20)<<setprecision(10)<<pos[
    0]+0.5+0.5*s_len<<" ";
479                         }
480                         else {//pointing right
481                                 out<<setw(20)<<setprecision(10)<<pos[
    0]+0.5-0.5*s_len<<" ";
482                         }
483                         out<<setw(20)<<setprecision(10)<<pos[1]<<" ";
484                         out<<setw(20)<<setprecision(10)<<dir[0]<<" ";
485                         out<<setw(20)<<setprecision(10)<<0.0<<endl;
486                 }
487         }
488 }
489
490 void Lattice::saturate(int& m_smcs){
491         //this function equilibrates the lattice in a saturation fiel
    d before returning to the previous field etc to equilibrate and take
    measurements
492
493         //swap the field and saturation field values
494         if(m_smcs>0) {
495                 vector<double>tempfield=field;
496                 field=satfield;
497                 for (int i=1; i<m_smcs; i++) {
498                         Mcupdate();
499                 }
500                 field=tempfield;
501         }
502 }
503
```

```
/* -*- Mode: C; indent-tabs-mode: t; c-basic-offset: 4; tab-width: 4 -*- */
/*
 * main.cc
 * Copyright (C) Adam Harman-Clarke 2010 <adam.harman-clarke@ucl.ac.uk>
 *
 * square_ice is free software: you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the
 * Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * square_ice is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>
#include <math.h>
#include <time.h>
#include "Lattice.h"
#include "Spin.h"
#include "stblib.h"
#include "Utilities.h"
#include "Globals.h"

using namespace std;

int main()
{
//initialise, read the input file for the parameters, init the rand num gen
        read_input();
        initialize_random();
        ofstream info ("info.out");
//get the parameters
        mcs = read_single_input_string<int>("mcs per field increment");
        int eqmcs = read_single_input_string<int>("equilibrium mcs");
        int satmcs = read_single_input_string<int>("saturation mcs");
        double tstart = read_single_input_string<double>("temperature at start");
        double tend = read_single_input_string<double>("temperature at finish");
        double tinc = read_single_input_string<double>("temperature increment");
        double bbig = read_single_input_string<double>("maximum field");
        double bsmall = read_single_input_string<double>("minimum field");
        double binc = read_single_input_string<double>("field increment");
        vector<double> fielddir = read_input_string<double>("field vector");
        double satmag = read_single_input_string<double>("sat field magnitude");
```

```
52          vector<double> satdir = read_input_string<double>("sat field directi
    on");
53          string intera = read_single_input_string<string>(string("intera
    ction type"));
54          string conf = read_single_input_string<string>(string("starting
    configuration"));
55  //calculate some useful quantities from the parameters, loop params,
    normalise the field etc
56          int tsteps;
57          if (tstart>tend) {
58                  tsteps=static_cast<int> ((tstart-tend)/tinc);
59                  tinc*=-1.;
60          }
61          else {
62                  tsteps=static_cast<int> ((tend-tstart)/tinc);
63          }
64          int bsteps=2*static_cast<int>((bbig-bsmall)/binc);
65          double denom=bsteps*(tsteps+1);
66          if (tsteps==0) denom=bsteps;
67          if (bsteps==0) denom=tsteps;
68          double prevpcdone=0.0;
69          double norm;
70          if (fielddir[0]==0. && fielddir[1]==0.) {
71                  norm=1.;
72          }
73          else {
74              norm=1./sqrt(fielddir[0]*fielddir[0]+fielddir[1]*fielddir
    [1]);
75          }
76          fielddir[0]*=norm;
77          fielddir[1]*=norm;
78          if (satdir[0]==0. && satdir[1]==0.) {
79                  norm=1.;
80          }
81          else {
82                  norm=1./sqrt(satdir[0]*satdir[0]+satdir[1]*satdir[1])
    ;
83          }
84          satdir[0]*=norm;
85          satdir[1]*=norm;
86  //initialise any quantities that need to be set to initialise the lat
    tice
87          temperature=tstart;
88          field=vec_const_mult(fielddir,bbig);
89          satfield=vec_const_mult(satdir,satmag);
90          exchj=exchint*pow((tcrit-temperature)/tcrit,beta);
91          time_t curtime;
92          struct tm *loctime;
93          /* Get the current time. */
94          curtime = time (NULL);
95          /* Convert it to local time representation. */
96          loctime = localtime (&curtime);
97          //define a lattice of spins
98          Lattice Latt;
99  //write some info to the info.txt file
100         info<<"SQUARE ICE SIMULATION"<<endl<<endl;
101         info<<"This simulation was started on "<<asctime(loctime)<<endl;
102         info<<"Number of spins in the lattice: "<<Latt.Size()<<endl<<endl;
103         info<<"Temperature ranged from "<<tstart<<" to "<<tend<<" in steps of "<<t
    inc<<endl<<endl;
104         info<<"Field magnitude ranged from "<<bbig<<" to "<<bsmall<<" in steps of "<
    <binc<<endl;
105         info<<"and the field direction was ("<<fielddir[0]<<", "<<fielddir[1]<<"
    )"<<endl<<endl;
106         info<<"At each temperature the lattice was equilibrated in a saturation field of "<<sat
```

```
mag<<"("<<satdir[0]<<", "<<satdir[1]<<") for "<<satmcs<<" Monte Carlo steps."<<endl<<endl;
107         info<<"The simulation performed "<<eqmcs<<" Monte Carlo steps to equilibrate the lattice"<<endl;
108         info<<"and "<<mcs<<" measurements with a Monte Carlo step in between at each new field/temp"<<endl<<endl;
109         info<<"The interaction energy model was "<<intera<<endl;
110         info<<"The starting spin configuration was "<<conf<<endl;
111         info<<endl<<"The simulation is..."<<endl;
112         //loop over temperature
113         for (int j=0; j<=tsteps; j++) {
114                 temperature=tstart+(j*tinc);
115                 //sweep from high to low to high field
116                 for (int k=0; k<=bsteps; k++) {
117                         if (k<=(bsteps/2)) {
118                                 field=vec_const_mult(fielddir,(bbig-(k*binc)));
119                         }
120                         else {
121                                 field=vec_const_mult(fielddir,(bsmall+((k-(bsteps/2))*binc)));
122                         }
123                         Latt.param_update();
124             Latt.saturate(satmcs);
125                         //equilibrate the lattice then do the reqd num of mc steps
126 //                      ofstream equ("mcequi.data");
127                         for (int i=0; i<eqmcs; i++) {
128 //                              cout<<"Equilibrating the lattice..."<<endl;
129                                 Latt.Mcupdate();
130 //                              if (i%100==0) {
131 //                                      equ<<setw(20)<<setprecision(10)<<i<<",";
132 //                                      equ<<setw(20)<<setprecision(10)<<run_ener<<endl;
133 //                              }
134                         }
135 //                      equ.close();
136                         for (int i=0; i<mcs; i++) {
137 //                              cout<<"Performing MC steps..."<<endl;
138                                 Latt.Mcupdate();
139                                 Latt.Measure();
140 //                              Latt.Lattfile();
141                         }
142                         //measure quantities and write them to a file
143                         Latt.Output();
144                 //      Latt.Lattfile();
145                         double numer=static_cast<double>((j*bsteps)+k);
146                         if(tsteps==0) numer=static_cast<double>(k);
147                         if(bsteps==0) numer=static_cast<double>(j);
148                         double pcdone=100.*numer/denom;
149                         if(floor(pcdone)-floor(prevpcdone)>0.5){
150                                 info<<setprecision(2)<<pcdone<<"% complete."<<endl;
151                                 info.flush();
152                         }
153                         prevpcdone=pcdone;
154                 }
155 //              Latt.Lattfile();
156         }
157         /* Get the current time. */
158         curtime = time (NULL);
159         /* Convert it to local time representation. */
```

```
160             loctime = localtime (&curtime);
161             info<<endl<<"This simulation finished on "<<asctime(loctime)<<endl;
162             return 0;
163     }
```

```
/**
 * @file SFMT.h
 *
 * @brief SIMD oriented Fast Mersenne Twister(SFMT) pseudorandom
 * number generator
 *
 * @author Mutsuo Saito (Hiroshima University)
 * @author Makoto Matsumoto (Hiroshima University)
 *
 * Copyright (C) 2006, 2007 Mutsuo Saito, Makoto Matsumoto and Hirosh
ima
 * University. All rights reserved.
 *
 * The new BSD License is applied to this software.
 * see LICENSE.txt
 *
 * @note We assume that your system has inttypes.h.  If your system
 * doesn't have inttypes.h, you have to typedef uint32_t and uint64_t
,
 * and you have to define PRIu64 and PRIx64 in this file as follows:
 * @verbatim
 typedef unsigned int uint32_t
 typedef unsigned long long uint64_t
 #define PRIu64 "llu"
 #define PRIx64 "llx"
@endverbatim
 * uint32_t must be exactly 32-bit unsigned integer type (no more, no
 * less), and uint64_t must be exactly 64-bit unsigned integer type.
 * PRIu64 and PRIx64 are used for printf function to print 64-bit
 * unsigned int and 64-bit unsigned int in hexadecimal format.
 */

#ifndef SFMT_H
#define SFMT_H

#include <stdio.h>

#if defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199901L)
  #include <inttypes.h>
#elif defined(_MSC_VER) || defined(__BORLANDC__)
  typedef unsigned int uint32_t;
  typedef unsigned __int64 uint64_t;
  #define inline __inline
#else
  #include <inttypes.h>
  #if defined(__GNUC__)
    #define inline __inline__
  #endif
#endif

#ifndef PRIu64
  #if defined(_MSC_VER) || defined(__BORLANDC__)
    #define PRIu64 "I64u"
    #define PRIx64 "I64x"
  #else
    #define PRIu64 "llu"
    #define PRIx64 "llx"
  #endif
#endif

#if defined(__GNUC__)
#define ALWAYSINLINE __attribute__((always_inline))
#else
#define ALWAYSINLINE
#endif
```

```
64
65  #if defined(_MSC_VER)
66    #if _MSC_VER >= 1200
67      #define PRE_ALWAYS __forceinline
68    #else
69      #define PRE_ALWAYS inline
70    #endif
71  #else
72    #define PRE_ALWAYS inline
73  #endif
74
75  uint32_t gen_rand32(void);
76  uint64_t gen_rand64(void);
77  void fill_array32(uint32_t *array, int size);
78  void fill_array64(uint64_t *array, int size);
79  void init_gen_rand(uint32_t seed);
80  void init_by_array(uint32_t *init_key, int key_length);
81  const char *get_idstring(void);
82  int get_min_array_size32(void);
83  int get_min_array_size64(void);
84
85  /* These real versions are due to Isaku Wada */
86  /** generates a random number on [0,1]-real-interval */
87  inline static double to_real1(uint32_t v)
88  {
89      return v * (1.0/4294967295.0);
90      /* divided by 2^32-1 */
91  }
92
93  /** generates a random number on [0,1]-real-interval */
94  inline static double genrand_real1(void)
95  {
96      return to_real1(gen_rand32());
97  }
98
99  /** generates a random number on [0,1)-real-interval */
100 inline static double to_real2(uint32_t v)
101 {
102     return v * (1.0/4294967296.0);
103     /* divided by 2^32 */
104 }
105
106 /** generates a random number on [0,1)-real-interval */
107 inline static double genrand_real2(void)
108 {
109     return to_real2(gen_rand32());
110 }
111
112 /** generates a random number on (0,1)-real-interval */
113 inline static double to_real3(uint32_t v)
114 {
115     return (((double)v) + 0.5)*(1.0/4294967296.0);
116     /* divided by 2^32 */
117 }
118
119 /** generates a random number on (0,1)-real-interval */
120 inline static double genrand_real3(void)
121 {
122     return to_real3(gen_rand32());
123 }
124 /** These real versions are due to Isaku Wada */
125
126 /** generates a random number on [0,1) with 53-bit resolution*/
127 inline static double to_res53(uint64_t v)
128 {
```

```
    return v * (1.0/18446744073709551616.0L);
}

/** generates a random number on [0,1) with 53-bit resolution from two
 * 32 bit integers */
inline static double to_res53_mix(uint32_t x, uint32_t y)
{
    return to_res53(x | ((uint64_t)y << 32));
}

/** generates a random number on [0,1) with 53-bit resolution
 */
inline static double genrand_res53(void)
{
    return to_res53(gen_rand64());
}

/** generates a random number on [0,1) with 53-bit resolution
    using 32bit integer.
 */
inline static double genrand_res53_mix(void)
{
    uint32_t x, y;

    x = gen_rand32();
    y = gen_rand32();
    return to_res53_mix(x, y);
}
#endif
```

```
/**
 * @file  SFMT.c
 * @brief SIMD oriented Fast Mersenne Twister(SFMT)
 *
 * @author Mutsuo Saito (Hiroshima University)
 * @author Makoto Matsumoto (Hiroshima University)
 *
 * Copyright (C) 2006,2007 Mutsuo Saito, Makoto Matsumoto and Hiroshi
ma
 * University. All rights reserved.
 *
 * The new BSD License is applied to this software, see LICENSE.txt
 */
#include <string.h>
#include <assert.h>
#include "SFMT.h"
#include "SFMT-params.h"

#if defined(__BIG_ENDIAN__) && !defined(__amd64) && !defined(BIG_ENDI
AN64)
#define BIG_ENDIAN64 1
#endif
#if defined(HAVE_ALTIVEC) && !defined(BIG_ENDIAN64)
#define BIG_ENDIAN64 1
#endif
#if defined(ONLY64) && !defined(BIG_ENDIAN64)
  #if defined(__GNUC__)
    #error "-DONLY64 must be specified with -DBIG_ENDIAN64"
  #endif
#undef ONLY64
#endif
/*------------------------------------------------------
  128-bit SIMD data type for Altivec, SSE2 or standard C
  ------------------------------------------------------*/
#if defined(HAVE_ALTIVEC)
  #if !defined(__APPLE__)
    #include <altivec.h>
  #endif
/** 128-bit data structure */
union W128_T {
    vector unsigned int s;
    uint32_t u[4];
};
/** 128-bit data type */
typedef union W128_T w128_t;

#elif defined(HAVE_SSE2)
  #include <emmintrin.h>

/** 128-bit data structure */
union W128_T {
    __m128i si;
    uint32_t u[4];
};
/** 128-bit data type */
typedef union W128_T w128_t;

#else

/** 128-bit data structure */
struct W128_T {
    uint32_t u[4];
};
/** 128-bit data type */
typedef struct W128_T w128_t;
```

```
64
65  #endif
66
67  /*--------------------------------------
68    FILE GLOBAL VARIABLES
69    internal state, index counter and flag
70    --------------------------------------*/
71  /** the 128-bit internal state array */
72  static w128_t sfmt[N];
73  /** the 32bit integer pointer to the 128-bit internal state array */
74  static uint32_t *psfmt32 = &sfmt[0].u[0];
75  #if !defined(BIG_ENDIAN64) || defined(ONLY64)
76  /** the 64bit integer pointer to the 128-bit internal state array */
77  static uint64_t *psfmt64 = (uint64_t *)&sfmt[0].u[0];
78  #endif
79  /** index counter to the 32-bit internal state array */
80  static int idx;
81  /** a flag: it is 0 if and only if the internal state is not yet
82   * initialized. */
83  static int initialized = 0;
84  /** a parity check vector which certificate the period of 2^{MEXP} */
85  static uint32_t parity[4] = {PARITY1, PARITY2, PARITY3, PARITY4};
86
87  /*----------------
88    STATIC FUNCTIONS
89    ----------------*/
90  inline static int idxof(int i);
91  inline static void rshift128(w128_t *out,  w128_t const *in, int shif
    t);
92  inline static void lshift128(w128_t *out,  w128_t const *in, int shif
    t);
93  inline static void gen_rand_all(void);
94  inline static void gen_rand_array(w128_t *array, int size);
95  inline static uint32_t func1(uint32_t x);
96  inline static uint32_t func2(uint32_t x);
97  static void period_certification(void);
98  #if defined(BIG_ENDIAN64) && !defined(ONLY64)
99  inline static void swap(w128_t *array, int size);
100 #endif
101
102 #if defined(HAVE_ALTIVEC)
103   #include "SFMT-alti.h"
104 #elif defined(HAVE_SSE2)
105   #include "SFMT-sse2.h"
106 #endif
107
108 /**
109  * This function simulate a 64-bit index of LITTLE ENDIAN
110  * in BIG ENDIAN machine.
111  */
112 #ifdef ONLY64
113 inline static int idxof(int i) {
114     return i ^ 1;
115 }
116 #else
117 inline static int idxof(int i) {
118     return i;
119 }
120 #endif
121 /**
122  * This function simulates SIMD 128-bit right shift by the standard C
    .
123  * The 128-bit integer given in in is shifted by (shift * 8) bits.
124  * This function simulates the LITTLE ENDIAN SIMD.
125  * @param out the output of this function
```

```
126  * @param in the 128-bit data to be shifted
127  * @param shift the shift value
128  */
129 #ifdef ONLY64
130 inline static void rshift128(w128_t *out, w128_t const *in, int shift
    ) {
131     uint64_t th, tl, oh, ol;
132 
133     th = ((uint64_t)in->u[2] << 32) | ((uint64_t)in->u[3]);
134     tl = ((uint64_t)in->u[0] << 32) | ((uint64_t)in->u[1]);
135 
136     oh = th >> (shift * 8);
137     ol = tl >> (shift * 8);
138     ol |= th << (64 - shift * 8);
139     out->u[0] = (uint32_t)(ol >> 32);
140     out->u[1] = (uint32_t)ol;
141     out->u[2] = (uint32_t)(oh >> 32);
142     out->u[3] = (uint32_t)oh;
143 }
144 #else
145 inline static void rshift128(w128_t *out, w128_t const *in, int shift
    ) {
146     uint64_t th, tl, oh, ol;
147 
148     th = ((uint64_t)in->u[3] << 32) | ((uint64_t)in->u[2]);
149     tl = ((uint64_t)in->u[1] << 32) | ((uint64_t)in->u[0]);
150 
151     oh = th >> (shift * 8);
152     ol = tl >> (shift * 8);
153     ol |= th << (64 - shift * 8);
154     out->u[1] = (uint32_t)(ol >> 32);
155     out->u[0] = (uint32_t)ol;
156     out->u[3] = (uint32_t)(oh >> 32);
157     out->u[2] = (uint32_t)oh;
158 }
159 #endif
160 /**
161  * This function simulates SIMD 128-bit left shift by the standard C.
162  * The 128-bit integer given in in is shifted by (shift * 8) bits.
163  * This function simulates the LITTLE ENDIAN SIMD.
164  * @param out the output of this function
165  * @param in the 128-bit data to be shifted
166  * @param shift the shift value
167  */
168 #ifdef ONLY64
169 inline static void lshift128(w128_t *out, w128_t const *in, int shift
    ) {
170     uint64_t th, tl, oh, ol;
171 
172     th = ((uint64_t)in->u[2] << 32) | ((uint64_t)in->u[3]);
173     tl = ((uint64_t)in->u[0] << 32) | ((uint64_t)in->u[1]);
174 
175     oh = th << (shift * 8);
176     ol = tl << (shift * 8);
177     oh |= tl >> (64 - shift * 8);
178     out->u[0] = (uint32_t)(ol >> 32);
179     out->u[1] = (uint32_t)ol;
180     out->u[2] = (uint32_t)(oh >> 32);
181     out->u[3] = (uint32_t)oh;
182 }
183 #else
184 inline static void lshift128(w128_t *out, w128_t const *in, int shift
    ) {
185     uint64_t th, tl, oh, ol;
186 
```

```
    th = ((uint64_t)in->u[3] << 32) | ((uint64_t)in->u[2]);
    tl = ((uint64_t)in->u[1] << 32) | ((uint64_t)in->u[0]);

    oh = th << (shift * 8);
    ol = tl << (shift * 8);
    oh |= tl >> (64 - shift * 8);
    out->u[1] = (uint32_t)(ol >> 32);
    out->u[0] = (uint32_t)ol;
    out->u[3] = (uint32_t)(oh >> 32);
    out->u[2] = (uint32_t)oh;
}
#endif

/**
 * This function represents the recursion formula.
 * @param r output
 * @param a a 128-bit part of the internal state array
 * @param b a 128-bit part of the internal state array
 * @param c a 128-bit part of the internal state array
 * @param d a 128-bit part of the internal state array
 */
#if (!defined(HAVE_ALTIVEC)) && (!defined(HAVE_SSE2))
#ifdef ONLY64
inline static void do_recursion(w128_t *r, w128_t *a, w128_t *b, w128
_t *c,
                                w128_t *d) {
    w128_t x;
    w128_t y;

    lshift128(&x, a, SL2);
    rshift128(&y, c, SR2);
    r->u[0] = a->u[0] ^ x.u[0] ^ ((b->u[0] >> SR1) & MSK2) ^ y.u[0]
        ^ (d->u[0] << SL1);
    r->u[1] = a->u[1] ^ x.u[1] ^ ((b->u[1] >> SR1) & MSK1) ^ y.u[1]
        ^ (d->u[1] << SL1);
    r->u[2] = a->u[2] ^ x.u[2] ^ ((b->u[2] >> SR1) & MSK4) ^ y.u[2]
        ^ (d->u[2] << SL1);
    r->u[3] = a->u[3] ^ x.u[3] ^ ((b->u[3] >> SR1) & MSK3) ^ y.u[3]
        ^ (d->u[3] << SL1);
}
#else
inline static void do_recursion(w128_t *r, w128_t *a, w128_t *b, w128
_t *c,
                                w128_t *d) {
    w128_t x;
    w128_t y;

    lshift128(&x, a, SL2);
    rshift128(&y, c, SR2);
    r->u[0] = a->u[0] ^ x.u[0] ^ ((b->u[0] >> SR1) & MSK1) ^ y.u[0]
        ^ (d->u[0] << SL1);
    r->u[1] = a->u[1] ^ x.u[1] ^ ((b->u[1] >> SR1) & MSK2) ^ y.u[1]
        ^ (d->u[1] << SL1);
    r->u[2] = a->u[2] ^ x.u[2] ^ ((b->u[2] >> SR1) & MSK3) ^ y.u[2]
        ^ (d->u[2] << SL1);
    r->u[3] = a->u[3] ^ x.u[3] ^ ((b->u[3] >> SR1) & MSK4) ^ y.u[3]
        ^ (d->u[3] << SL1);
}
#endif
#endif

#if (!defined(HAVE_ALTIVEC)) && (!defined(HAVE_SSE2))
/**
 * This function fills the internal state array with pseudorandom
 * integers.
```

```
 */
inline static void gen_rand_all(void) {
    int i;
    w128_t *r1, *r2;

    r1 = &sfmt[N - 2];
    r2 = &sfmt[N - 1];
    for (i = 0; i < N - POS1; i++) {
        do_recursion(&sfmt[i], &sfmt[i], &sfmt[i + POS1], r1, r2);
        r1 = r2;
        r2 = &sfmt[i];
    }
    for (; i < N; i++) {
        do_recursion(&sfmt[i], &sfmt[i], &sfmt[i + POS1 - N], r1, r2)
;
        r1 = r2;
        r2 = &sfmt[i];
    }
}

/**
 * This function fills the user-specified array with pseudorandom
 * integers.
 *
 * @param array an 128-bit array to be filled by pseudorandom numbers
.
 * @param size number of 128-bit pseudorandom numbers to be generated
.
 */
inline static void gen_rand_array(w128_t *array, int size) {
    int i, j;
    w128_t *r1, *r2;

    r1 = &sfmt[N - 2];
    r2 = &sfmt[N - 1];
    for (i = 0; i < N - POS1; i++) {
        do_recursion(&array[i], &sfmt[i], &sfmt[i + POS1], r1, r2);
        r1 = r2;
        r2 = &array[i];
    }
    for (; i < N; i++) {
        do_recursion(&array[i], &sfmt[i], &array[i + POS1 - N], r1, r
2);
        r1 = r2;
        r2 = &array[i];
    }
    for (; i < size - N; i++) {
        do_recursion(&array[i], &array[i - N], &array[i + POS1 - N],
r1, r2);
        r1 = r2;
        r2 = &array[i];
    }
    for (j = 0; j < 2 * N - size; j++) {
        sfmt[j] = array[j + size - N];
    }
    for (; i < size; i++, j++) {
        do_recursion(&array[i], &array[i - N], &array[i + POS1 - N],
r1, r2);
        r1 = r2;
        r2 = &array[i];
        sfmt[j] = array[i];
    }
}
#endif

```

```
#if defined(BIG_ENDIAN64) && !defined(ONLY64) && !defined(HAVE_ALTIVE
C)
inline static void swap(w128_t *array, int size) {
    int i;
    uint32_t x, y;

    for (i = 0; i < size; i++) {
        x = array[i].u[0];
        y = array[i].u[2];
        array[i].u[0] = array[i].u[1];
        array[i].u[2] = array[i].u[3];
        array[i].u[1] = x;
        array[i].u[3] = y;
    }
}
#endif
/**
 * This function represents a function used in the initialization
 * by init_by_array
 * @param x 32-bit integer
 * @return 32-bit integer
 */
static uint32_t func1(uint32_t x) {
    return (x ^ (x >> 27)) * (uint32_t)1664525UL;
}

/**
 * This function represents a function used in the initialization
 * by init_by_array
 * @param x 32-bit integer
 * @return 32-bit integer
 */
static uint32_t func2(uint32_t x) {
    return (x ^ (x >> 27)) * (uint32_t)1566083941UL;
}

/**
 * This function certificate the period of 2^{MEXP}
 */
static void period_certification(void) {
    int inner = 0;
    int i, j;
    uint32_t work;

    for (i = 0; i < 4; i++)
        inner ^= psfmt32[idxof(i)] & parity[i];
    for (i = 16; i > 0; i >>= 1)
        inner ^= inner >> i;
    inner &= 1;
    /* check OK */
    if (inner == 1) {
        return;
    }
    /* check NG, and modification */
    for (i = 0; i < 4; i++) {
        work = 1;
        for (j = 0; j < 32; j++) {
            if ((work & parity[i]) != 0) {
                psfmt32[idxof(i)] ^= work;
                return;
            }
            work = work << 1;
        }
    }
}
```

```
373
374 /*----------------
375   PUBLIC FUNCTIONS
376   ----------------*/
377 /**
378  * This function returns the identification string.
379  * The string shows the word size, the Mersenne exponent,
380  * and all parameters of this generator.
381  */
382 const char *get_idstring(void) {
383     return IDSTR;
384 }
385
386 /**
387  * This function returns the minimum size of array used for \b
388  * fill_array32() function.
389  * @return minimum size of array used for fill_array32() function.
390  */
391 int get_min_array_size32(void) {
392     return N32;
393 }
394
395 /**
396  * This function returns the minimum size of array used for \b
397  * fill_array64() function.
398  * @return minimum size of array used for fill_array64() function.
399  */
400 int get_min_array_size64(void) {
401     return N64;
402 }
403
404 #ifndef ONLY64
405 /**
406  * This function generates and returns 32-bit pseudorandom number.
407  * init_gen_rand or init_by_array must be called before this function
    .
408  * @return 32-bit pseudorandom number
409  */
410 uint32_t gen_rand32(void) {
411     uint32_t r;
412
413     assert(initialized);
414     if (idx >= N32) {
415         gen_rand_all();
416         idx = 0;
417     }
418     r = psfmt32[idx++];
419     return r;
420 }
421 #endif
422 /**
423  * This function generates and returns 64-bit pseudorandom number.
424  * init_gen_rand or init_by_array must be called before this function
    .
425  * The function gen_rand64 should not be called after gen_rand32,
426  * unless an initialization is again executed.
427  * @return 64-bit pseudorandom number
428  */
429 uint64_t gen_rand64(void) {
430 #if defined(BIG_ENDIAN64) && !defined(ONLY64)
431     uint32_t r1, r2;
432 #else
433     uint64_t r;
434 #endif
435
```

```
436     assert(initialized);
437     assert(idx % 2 == 0);
438 
439     if (idx >= N32) {
440         gen_rand_all();
441         idx = 0;
442     }
443 #if defined(BIG_ENDIAN64) && !defined(ONLY64)
444     r1 = psfmt32[idx];
445     r2 = psfmt32[idx + 1];
446     idx += 2;
447     return ((uint64_t)r2 << 32) | r1;
448 #else
449     r = psfmt64[idx / 2];
450     idx += 2;
451     return r;
452 #endif
453 }
454 
455 #ifndef ONLY64
456 /**
457  * This function generates pseudorandom 32-bit integers in the
458  * specified array[] by one call. The number of pseudorandom integers
459  * is specified by the argument size, which must be at least 624 and
    a
460  * multiple of four.  The generation by this function is much faster
461  * than the following gen_rand function.
462  *
463  * For initialization, init_gen_rand or init_by_array must be called
464  * before the first call of this function. This function can not be
465  * used after calling gen_rand function, without initialization.
466  *
467  * @param array an array where pseudorandom 32-bit integers are fille
    d
468  * by this function.  The pointer to the array must be \b "aligned"
469  * (namely, must be a multiple of 16) in the SIMD version, since it
470  * refers to the address of a 128-bit integer.  In the standard C
471  * version, the pointer is arbitrary.
472  *
473  * @param size the number of 32-bit pseudorandom integers to be
474  * generated.  size must be a multiple of 4, and greater than or equa
    l
475  * to (MEXP / 128 + 1) * 4.
476  *
477  * @note \b memalign or \b posix_memalign is available to get aligned
478  * memory. Mac OSX doesn't have these functions, but \b malloc of OSX
479  * returns the pointer to the aligned memory block.
480  */
481 void fill_array32(uint32_t *array, int size) {
482     assert(initialized);
483     assert(idx == N32);
484     assert(size % 4 == 0);
485     assert(size >= N32);
486 
487     gen_rand_array((w128_t *)array, size / 4);
488     idx = N32;
489 }
490 #endif
491 
492 /**
493  * This function generates pseudorandom 64-bit integers in the
494  * specified array[] by one call. The number of pseudorandom integers
495  * is specified by the argument size, which must be at least 312 and
    a
496  * multiple of two.  The generation by this function is much faster
```

```
 * than the following gen_rand function.
 *
 * For initialization, init_gen_rand or init_by_array must be called
 * before the first call of this function. This function can not be
 * used after calling gen_rand function, without initialization.
 *
 * @param array an array where pseudorandom 64-bit integers are fille
d
 * by this function.  The pointer to the array must be "aligned"
 * (namely, must be a multiple of 16) in the SIMD version, since it
 * refers to the address of a 128-bit integer.  In the standard C
 * version, the pointer is arbitrary.
 *
 * @param size the number of 64-bit pseudorandom integers to be
 * generated.  size must be a multiple of 2, and greater than or equa
l
 * to (MEXP / 128 + 1) * 2
 *
 * @note \b memalign or \b posix_memalign is available to get aligned
 * memory. Mac OSX doesn't have these functions, but \b malloc of OSX
 * returns the pointer to the aligned memory block.
 */
void fill_array64(uint64_t *array, int size) {
    assert(initialized);
    assert(idx == N32);
    assert(size % 2 == 0);
    assert(size >= N64);

    gen_rand_array((w128_t *)array, size / 2);
    idx = N32;

#if defined(BIG_ENDIAN64) && !defined(ONLY64)
    swap((w128_t *)array, size /2);
#endif
}

/**
 * This function initializes the internal state array with a 32-bit
 * integer seed.
 *
 * @param seed a 32-bit integer used as the seed.
 */
void init_gen_rand(uint32_t seed) {
    int i;

    psfmt32[idxof(0)] = seed;
    for (i = 1; i < N32; i++) {
        psfmt32[idxof(i)] = 1812433253UL * (psfmt32[idxof(i - 1)]
                                            ^ (psfmt32[idxof(i - 1)]
>> 30))
              + i;
    }
    idx = N32;
    period_certification();
    initialized = 1;
}

/**
 * This function initializes the internal state array,
 * with an array of 32-bit integers used as the seeds
 * @param init_key the array of 32-bit integers, used as a seed.
 * @param key_length the length of init_key.
 */
void init_by_array(uint32_t *init_key, int key_length) {
    int i, j, count;
```

```
    uint32_t r;
    int lag;
    int mid;
    int size = N * 4;

    if (size >= 623) {
        lag = 11;
    } else if (size >= 68) {
        lag = 7;
    } else if (size >= 39) {
        lag = 5;
    } else {
        lag = 3;
    }
    mid = (size - lag) / 2;

    memset(sfmt, 0x8b, sizeof(sfmt));
    if (key_length + 1 > N32) {
        count = key_length + 1;
    } else {
        count = N32;
    }
    r = func1(psfmt32[idxof(0)] ^ psfmt32[idxof(mid)]
              ^ psfmt32[idxof(N32 - 1)]);
    psfmt32[idxof(mid)] += r;
    r += key_length;
    psfmt32[idxof(mid + lag)] += r;
    psfmt32[idxof(0)] = r;

    count--;
    for (i = 1, j = 0; (j < count) && (j < key_length); j++) {
        r = func1(psfmt32[idxof(i)] ^ psfmt32[idxof((i + mid) % N32)]
                  ^ psfmt32[idxof((i + N32 - 1) % N32)]);
        psfmt32[idxof((i + mid) % N32)] += r;
        r += init_key[j] + i;
        psfmt32[idxof((i + mid + lag) % N32)] += r;
        psfmt32[idxof(i)] = r;
        i = (i + 1) % N32;
    }
    for (; j < count; j++) {
        r = func1(psfmt32[idxof(i)] ^ psfmt32[idxof((i + mid) % N32)]
                  ^ psfmt32[idxof((i + N32 - 1) % N32)]);
        psfmt32[idxof((i + mid) % N32)] += r;
        r += i;
        psfmt32[idxof((i + mid + lag) % N32)] += r;
        psfmt32[idxof(i)] = r;
        i = (i + 1) % N32;
    }
    for (j = 0; j < N32; j++) {
        r = func2(psfmt32[idxof(i)] + psfmt32[idxof((i + mid) % N32)]
                  + psfmt32[idxof((i + N32 - 1) % N32)]);
        psfmt32[idxof((i + mid) % N32)] ^= r;
        r -= i;
        psfmt32[idxof((i + mid + lag) % N32)] ^= r;
        psfmt32[idxof(i)] = r;
        i = (i + 1) % N32;
    }

    idx = N32;
    period_certification();
    initialized = 1;
}
```

```
#ifndef SFMT_PARAMS_H
#define SFMT_PARAMS_H

#if !defined(MEXP)
#ifdef __GNUC__
  #warning "MEXP is not defined. I assume MEXP is 19937."
#endif
  #define MEXP 19937
#endif
/*------------------
  BASIC DEFINITIONS
  ------------------*/
/** Mersenne Exponent. The period of the sequence
 *  is a multiple of 2^MEXP-1.
 * #define MEXP 19937 */
/** SFMT generator has an internal state array of 128-bit integers,
 * and N is its size. */
#define N (MEXP / 128 + 1)
/** N32 is the size of internal state array when regarded as an array
 * of 32-bit integers.*/
#define N32 (N * 4)
/** N64 is the size of internal state array when regarded as an array
 * of 64-bit integers.*/
#define N64 (N * 2)

/*----------------------
  the parameters of SFMT
  following definitions are in paramsXXXX.h file.
  ----------------------*/
/** the pick up position of the array.
#define POS1 122
*/

/** the parameter of shift left as four 32-bit registers.
#define SL1 18
 */

/** the parameter of shift left as one 128-bit register.
 * The 128-bit integer is shifted by (SL2 * 8) bits.
#define SL2 1
*/

/** the parameter of shift right as four 32-bit registers.
#define SR1 11
*/

/** the parameter of shift right as one 128-bit register.
 * The 128-bit integer is shifted by (SL2 * 8) bits.
#define SR2 1
*/

/** A bitmask, used in the recursion.  These parameters are introduce
d
 * to break symmetry of SIMD.
#define MSK1 0xdfffffefU
#define MSK2 0xddfecb7fU
#define MSK3 0xbffaffffU
#define MSK4 0xbffffff6U
*/

/** These definitions are part of a 128-bit period certification vect
or.
#define PARITY1 0x00000001U
#define PARITY2 0x00000000U
#define PARITY3 0x00000000U
```

```
64 #define PARITY4 0xc98e126aU
65 */
66
67 #if MEXP == 607
68   #include "SFMT-params607.h"
69 #elif MEXP == 1279
70   #include "SFMT-params1279.h"
71 #elif MEXP == 2281
72   #include "SFMT-params2281.h"
73 #elif MEXP == 4253
74   #include "SFMT-params4253.h"
75 #elif MEXP == 11213
76   #include "SFMT-params11213.h"
77 #elif MEXP == 19937
78   #include "SFMT-params19937.h"
79 #elif MEXP == 44497
80   #include "SFMT-params44497.h"
81 #elif MEXP == 86243
82   #include "SFMT-params86243.h"
83 #elif MEXP == 132049
84   #include "SFMT-params132049.h"
85 #elif MEXP == 216091
86   #include "SFMT-params216091.h"
87 #else
88 #ifdef __GNUC__
89   #error "MEXP is not valid."
90   #undef MEXP
91 #else
92   #undef MEXP
93 #endif
94
95 #endif
96
97 #endif /* SFMT_PARAMS_H */
```

```
#ifndef SFMT_PARAMS19937_H
#define SFMT_PARAMS19937_H

#define POS1    122
#define SL1     18
#define SL2     1
#define SR1     11
#define SR2     1
#define MSK1    0xdfffffefU
#define MSK2    0xddfecb7fU
#define MSK3    0xbffaffffU
#define MSK4    0xbffffff6U
#define PARITY1 0x00000001U
#define PARITY2 0x00000000U
#define PARITY3 0x00000000U
#define PARITY4 0x13c9e684U


/* PARAMETERS FOR ALTIVEC */
#if defined(__APPLE__)  /* For OSX */
    #define ALTI_SL1    (vector unsigned int)(SL1, SL1, SL1, SL1)
    #define ALTI_SR1    (vector unsigned int)(SR1, SR1, SR1, SR1)
    #define ALTI_MSK    (vector unsigned int)(MSK1, MSK2, MSK3, MSK4)
    #define ALTI_MSK64 \
        (vector unsigned int)(MSK2, MSK1, MSK4, MSK3)
    #define ALTI_SL2_PERM \
        (vector unsigned char)(1,2,3,23,5,6,7,0,9,10,11,4,13,14,15,8)
    #define ALTI_SL2_PERM64 \
        (vector unsigned char)(1,2,3,4,5,6,7,31,9,10,11,12,13,14,15,0
)
    #define ALTI_SR2_PERM \
        (vector unsigned char)(7,0,1,2,11,4,5,6,15,8,9,10,17,12,13,14
)
    #define ALTI_SR2_PERM64 \
        (vector unsigned char)(15,0,1,2,3,4,5,6,17,8,9,10,11,12,13,14
)
#else   /* For OTHER OSs(Linux?) */
    #define ALTI_SL1    {SL1, SL1, SL1, SL1}
    #define ALTI_SR1    {SR1, SR1, SR1, SR1}
    #define ALTI_MSK    {MSK1, MSK2, MSK3, MSK4}
    #define ALTI_MSK64  {MSK2, MSK1, MSK4, MSK3}
    #define ALTI_SL2_PERM       {1,2,3,23,5,6,7,0,9,10,11,4,13,14,15,
8}
    #define ALTI_SL2_PERM64     {1,2,3,4,5,6,7,31,9,10,11,12,13,14,15
,0}
    #define ALTI_SR2_PERM       {7,0,1,2,11,4,5,6,15,8,9,10,17,12,13,
14}
    #define ALTI_SR2_PERM64     {15,0,1,2,3,4,5,6,17,8,9,10,11,12,13,
14}
#endif  /* For OSX */
#define IDSTR   "SFMT-19937:122-18-1-11-1:dfffffef-ddfecb7f-bffaffff-bffffff6"

#endif /* SFMT_PARAMS19937_H */
```

```
/* The Spin class
 *
 * A definition of a spin for use in magnetic simulations
 */

#ifndef SPIN_H
#define SPIN_H

#include<vector>

using namespace std;

class Spin {
private:
        vector<double> m_pos;
        vector<double> m_dir;
        static int idgen;
        int name;
public:
        Spin(double posx, double posy, double dirx, double diry);
        void Setpos(double posx, double posy);
        void Setdir(double dirx, double diry);
        vector<double> Getpos () {return m_pos;}
        vector<double> Getdir () {return m_dir;}
        void Flipspin();
        int Id() {return name;}
};

#endif
```

```
/*
 *  Spin.cpp
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 08/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */


#include "Spin.h"
#include <iostream>

int Spin::idgen=0;

//Constructor
Spin::Spin(double posx, double posy, double dirx, double diry){
        name = idgen++;
        Setpos(posx, posy);
        Setdir(dirx, diry);
}

/*
 *---  Member functions
 */

//Clear the position vector and assign new x and y values
void Spin::Setpos(double posx, double posy){
        m_pos.clear();
        m_pos.push_back(posx);
        m_pos.push_back(posy);
}

//Clear the spin direction vector and assign new x and y values
void Spin::Setdir(double dirx, double diry){
        m_dir.clear();
        m_dir.push_back(dirx);
        m_dir.push_back(diry);
}

//Flip the spin direction vector
void Spin::Flipspin(){
        double oldy=m_dir.back();
        m_dir.pop_back();
        double oldx=m_dir.back();
        m_dir.clear();
        m_dir.push_back(-oldx);
        m_dir.push_back(-oldy);
}
```

```
/*************************************************************************
    Copyright (C) 2007 by Simon Banks
    University of Oxford, UCL   and the London Centre for Nanotechnology
                email: simon.banks@chem.ox.ac.uk

    $Revision: 1.5 $
    $Author: simon $
    $Date: 2009/07/22 13:01:22 $

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the
    Free Software Foundation, Inc.,
    59 Temple Place − Suite 330, Boston, MA  02111−1307, USA.
 *************************************************************************/
#ifndef STBLIB_H
#define STBLIB_H
#include "SFMT.h"
#include <cmath>
#include <vector>
#include <iostream>
#include <time.h>
#include <string>
#include <cstdlib>

// Return val1 with prob 1/2, else return val2
int prob(const int& val1, const int& val2, const double& probability);
char prob(const char& val1, const char& val2, const double& probability);

// Print string function for use in debugging
void ps(std::string& s);

// Quick function to print out message during testing
inline static void ptt() {
        std::cout << "Made it this far!\n";
```

```
};

// Initialize random number generator:
inline static void initialize_random() {
        srand(time(NULL));
        int seed = rand();
        init_gen_rand(seed); // Using SFMT generator.
};

// Does the obvious!
template<typename T>
void swap_val(const T& val1, const T& val2, T& target) {
        if (target == val1) target = val2;
        else if (target == val2) target = val1;
        else std::cerr << "Error in swap_val --> target not equal to either value!" <<
std::endl;
}

// Return an integer from 1 to pmax with equal probability
inline int equal_prob_ints(const int& pmax) {
        return (int)ceil(pmax*genrand_real1()); // genrand_real1() gi
ves rand num on interval [0,1]
};

// Inner product of two vectors.
template<typename T>
inline T dot(const std::vector <T>& vec1, const std::vector <T>& vec2
) {
        double sum = 0;
        if (vec1.size() != vec2.size()) std::cerr << "Error in dot --> vec1
and vec2 are different lengths!\n";
        else {
                int vsize = vec1.size();
                for (int i=0; i<vsize; i++) sum += vec1[i]*vec2[i];
        }
        return sum;
};


// Return the modulus of a vector.
template<typename T>
inline double modvec(const std::vector <T>& vec) {
        return (sqrt(dot(vec,vec)));
};

// Random number generator on interval [0,1]. For different intervals
 look at SFMT.h for examples
inline static double stb_rand() {
        return genrand_real1(); // Gives random number on interval [0
,1]
};

// Add two vectors.
template<typename T>
std::vector<T> vec_add(const std::vector<T>& vec1, const std::vector<
T>& vec2) {
        std::vector <T> vecsum(vec1.size(),0);
        if (vec1.size() != vec2.size()) std::cerr << "Error in vec_add -->
vec1 and vec2 are different lengths!\n";
        else {
                int vsize = vec1.size();
                for (int i=0; i<vsize; i++) {
                        vecsum[i] = vec1[i] + vec2[i];
                }
        }
```

```
102         return vecsum;
103 };
104
105 // Subtract two vectors.
106 template<typename T>
107 std::vector<T> vec_sub(const std::vector<T>& vec1, const std::vector<
    T>& vec2) {
108         std::vector <double> vecsum(vec1.size(),0);
109         if (vec1.size() != vec2.size()) std::cerr << "Error in vec_sub -->
     vec1 and vec2 are different lengths!\n";
110         else {
111                 int vsize = vec1.size();
112                 for (int i=0; i<vsize; i++) {
113                         vecsum[i] = vec1[i] - vec2[i];
114                 }
115         }
116         return vecsum;
117 };
118
119 // Multiply a vector by a constant.
120 template<typename T, typename T2>
121 std::vector<T> vec_const_mult(const std::vector<T>& vec, const T2& c)
     {
122         int vsize = vec.size();
123         std::vector<double> newvec(vsize,0);
124         for (int i=0; i<vsize; i++) {
125                 newvec[i] = vec[i]*c;
126         }
127         return newvec;
128 };
129
130 template<typename T>
131 inline int stb_round(T num) {
132         return static_cast<int>(num+0.5);
133 };
134
135 // Determine the cumulants and susceptibility of a given array of dat
    a
136 std::vector<double> cumulants(std::vector<double>& data_array, const
    std::string& MC_observable,
137
                                                        const int& N, const
     double& T, const bool& pdf);
138
139 // Generate a pdf for an array of data. Called from "cumulants".
140 void generate_pdf(const std::vector<double>& raw_data_array, const st
    d::string& MC_observable,
141
    const double& ave, const double& sigma);
142
143
144 #endif
```

```
/*************************************************************************
    Copyright (C) 2007 by Simon Banks
    University of Oxford, UCL   and the London Centre for Nanotechnology
                email: simon.banks@chem.ox.ac.uk

    $Revision: 1.4 $
    $Author: simon $
    $Date: 2009/07/22 13:01:22 $

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the
    Free Software Foundation, Inc.,
    59 Temple Place − Suite 330, Boston, MA  02111−1307, USA.
 *************************************************************************/
#include "stblib.h"
#include "SFMT.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <time.h>
#include <algorithm>

using namespace std;

inline int my_round(double x)
{
        return static_cast<int>(x > 0.0 ? x + 0.5 : x − 0.5);
}

int prob(const int& val1, const int& val2, const double& probability)
 { // Return one of two vals with prob 1/2
        if (stb_rand() < 0.5) return val1;
        else return val2;
}

```

```
char prob(const char& val1, const char& val2, const double& probabili
ty) {
        if (stb_rand() < 0.5) return val1;
        else return val2;
}

void ps(string& s) {
        cout << s << endl;
};

vector<double> cumulants(vector<double>& data_array, const string& MC
_observable,

                         const int& N, const double& T, const bool& p
df) {
        string output_filename = MC_observable + "_cumulants";
        ofstream output(output_filename.c_str(),ios::app);

        double a1 = 0.0;
        double a2 = 0.0;
        double a3 = 0.0;
        double a4 = 0.0;
        double c1 = 0.0;
        double c2 = 0.0;

        int num_data = data_array.size();
        vector<double> intensive_data_array(num_data,0);

        for (int i=0; i<num_data; i++) {
                intensive_data_array[i] = data_array[i]/(double)N;
                double x = intensive_data_array[i];
                a1 += x;
                a2 += x*x;
                a3 += x*x*x;
                a4 += x*x*x*x;
        }

        a1 = a1/(double)num_data;
        a2 = a2/(double)num_data;
        a3 = a3/(double)num_data;
        a4 = a4/(double)num_data;

        vector<double> cumulant_vec(4,0);

        cumulant_vec[0] = a1;
        c1 = a1;
        cumulant_vec[1] = sqrt(a2-a1*a1);
        c2 = cumulant_vec[1];
        cumulant_vec[2] = (a3 - 3*a2*a1 + 2*a1*a1*a1)/(c2*c2*c2);
        cumulant_vec[3] = (a4 - 4*a3*a1 - 3*a2*a2 + 12*a2*a1*a1 - 6*a
1*a1*a1*a1)/(c2*c2*c2*c2);

        output << "\n\nT = " << T << "\nN = " << N
                        << "\nc1\t" << cumulant_vec[0]
                        << "\nc2\t" << cumulant_vec[1]
                        << "\nc3\t" << cumulant_vec[2]
                        << "\nc4\t" << cumulant_vec[3]

                        << "\nBinder\t" << 1-a4/(3*a2*a2)
                        << "\nC(T) =\t" << N*c2*c2/(T*T)
                        << "\nChi(T) =\t"<< N*c2*c2/T << endl;

        output.close();

        if (pdf) generate_pdf(intensive_data_array,MC_observable,c1,c
```

```
2);

        return cumulant_vec;

};

void generate_pdf(const vector<double>& raw_data_array, const string& MC_observable,

const double& ave, const double& sigma) {

        ////////////////////////////////////////////////////////////////
  // Arranging the contents of data_array and binning to get
  // probabilities
        ////////////////////////////////////////////////////////////////

        int number_of_bins = 1;                                 // The total number of data bins
        double lower_bound = 0.0;                               // floor (lowest mag recorded)
        double upper_bound = 0.0;                               // ceil (highest mag recorded)
        double interval1 = 0.0;                                 // bin range in region 1 (m<-2)
        double interval2 = 0.0;                                 // bin range in region 2 (-2<m<4)
        double interval3 = 0.0;                                 // bin range in region 3 (4<m<6)
        double interval4 = 0.0;                                 // bin range in region 4 (6<m)
        int bin_num = 0;                                        // indexing variable
        vector<double> data_array = raw_data_array;
        int data_array_size = data_array.size();                  // Number of m values recorded.
        double mark1 = 0.0;
        double mark2 = 0.0;                                       // Region separation points
        double mark3 = 0.0;

        string output_filename = MC_observable + "_pdf";
        ofstream probs(output_filename.c_str(), std::ios::app);
        ifstream binin("bin_details");

        ////////////////////////////////////////////////////////////////
  // Normalize and shift origin of m to (m-<m>)/sigma. Then sort
  // array of values into ascending order.
        ////////////////////////////////////////////////////////////////

        for (int i=0; i<data_array_size; i++) data_array[i] = (data_array[i]-ave)/sigma;
        sort(data_array.begin(), data_array.end());

        ////////////////////////////////////////////////////////////////
  // Calculate number of bins in each region and the overall total
        ////////////////////////////////////////////////////////////////

        lower_bound = floor(data_array[0]);
        upper_bound = ceil(data_array[data_array_size-1]);
```

```
151         binin >> mark1 >> mark2 >> mark3 >> interval1 >> interval2 >>
     interval3 >> interval4;
152 
153         int bin_num1=0,bin_num2=0,bin_num3=0,bin_num4=0;
154 
155         bin_num1 = (int)round((mark1-lower_bound)/interval1);
156         bin_num2 = (int)round((mark2-mark1)/interval2);
157         bin_num3 = (int)round((mark3-mark2)/interval3);
158         bin_num4 = (int)round((upper_bound-mark3)/interval4);
159 
160         double dnumber_of_bins = bin_num1 + bin_num2 + bin_num3 + bin
    _num4 + 0.1;
161         number_of_bins = (int)dnumber_of_bins;
162         cout<<number_of_bins<<endl;
163 
164         vector < double > bin_array(number_of_bins,0.0);            /
    / Contains number of occurences of
165                                                                     // m in
     each bin.
166         vector < double > bin_values(number_of_bins,0.0);           /
    / Upper value in each bin
167 
168         ////////////////////////////////////////////////////////////
    /////////
169   // Assign the ranges associated with each bin
170         ////////////////////////////////////////////////////////////
    /////////
171 cout<<lower_bound<<endl;
172         bin_values[0] = lower_bound;
173 
174         for (int i=1; i<number_of_bins; i++)
175         {
176                 if (i<=bin_num1) bin_values[i] = bin_values[i-1] + in
    terval1;
177                 if (i>bin_num1 && i<=(bin_num1+bin_num2)) bin_values[
    i] = bin_values[i-1] + interval2;
178                 if (i>(bin_num1+bin_num2) && i<=(bin_num1+bin_num2+bi
    n_num3)) bin_values[i]=bin_values[i-1]+interval3;
179                 if (i>(bin_num1+bin_num2+bin_num3)) bin_values[i] = b
    in_values[i-1] + interval4;
180         }
181         ////////////////////////////////////////////////////////////
    /////////
182   // Evaluate the number of occurences of m in each bin -
183   // this is made considerably easier by having sorted data_array
184         ////////////////////////////////////////////////////////////
    /////////
185 
186         for (int i=0; i<data_array_size; i++)
187         {
188                 if (data_array[i] > bin_values[number_of_bins-1])
    // i.e. highest bin effectively unbounded
189                 {
190                         bin_array[number_of_bins-1]++;
191                         continue;
192                 }
193                 if (data_array[i] <= bin_values[bin_num]) bin_array[b
    in_num]++;
194                 else
195                 {
196                         i--;
197                         bin_num++;
198                 }
199         }
200         ////////////////////////////////////////////////////////////
```

```
//////
   // Shift the values of bin_values to the centre of each bin range
   // for correct plotting of the distribution function. Also, the
   // values of bin_array are normalized to give probabilities, note
   // normalization depends on the width of the bin (i.e. the region)
      ////////////////////////////////////////////////////////
//////

        for (int i=0; i<number_of_bins; i++)
        {
                if (i<=bin_num1)
                {
                        bin_values[i] -= 0.5*interval1;
                        bin_array[i] = bin_array[i]/(data_array.size(
)*interval1);
                }
                else if (i>bin_num1 && i<=(bin_num1+bin_num2))
                {
                        bin_values[i] -= 0.5*interval2;
                        bin_array[i] = bin_array[i]/(data_array.size(
)*interval2);
                }
                else if (i>(bin_num1+bin_num2) && i<=(bin_num1+bin_nu
m2+bin_num3))
                {
                        bin_values[i] -= 0.5*interval3;
                        bin_array[i] = bin_array[i]/(data_array.size(
)*interval3);
                }
                else if (i>(bin_num1+bin_num2+bin_num3))
                {
                        bin_values[i] -= 0.5*interval4;
                        bin_array[i] = bin_array[i]/(data_array.size(
)*interval4);
                }
                probs << bin_values[i] << "\t" << bin_array[i] << "\n"
;
        }
        probs << "###############\n" ;
        probs.close();
}
```

```
/*
 *  Utilities.h
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 09/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#ifndef UTILITIES_H
#define UTILITIES_H

#include <vector>
#include <string>
#include <sstream>
#include <iostream>

void read_input();
std::string search_input(const std::string&);
std::string s_top(const std::string& s);

// For reading from the input file
template<typename T>
std::vector<T> read_input_string(const std::string& ref) {
        std::vector<T> input_data;
        T value;
        std::string data;
        data = search_input(ref);

        std::istringstream ss_data(data);
        while (ss_data >> value) input_data.push_back(value);

        return input_data;
};

template<typename T>
T read_single_input_string(const std::string& ref) {
        T value;
        std::string data;
        data = search_input(ref);
        std::istringstream ss_data(data);
        ss_data >> value;

        return value;
};

#endif
```

```
/*
 *  Utilities.cpp
 *  sq_ice
 *
 *  Created by Adam Harman-Clarke on 09/11/2010.
 *  Copyright 2010 Adam Harman-Clarke. All rights reserved.
 *
 */

#include "Utilities.h"
#include <ctype.h>
#include <string>
#include <iostream>
#include <fstream>
#include <vector>


using namespace std;

const char* input_file_name="userinputs.ipt";
vector<string> input_string;

// Read the input file as a vector of strings
void read_input() {
        string line;
        ifstream input("userinputs.ipt");
        while (getline(input,line,'#')) {
                for (int i=0; i<line.size(); i++) {
                        line[i] = tolower(line[i]);          // Conver
t all characters to lower case (requires <ctype.h>)
                }
                input_string.push_back(line);
        }
};

// Read up to the first : in a string
string s_top(const string& s) {
        int i = s.find(":");
        return(s.substr(0,i));
};

// Read from the first : in a string
string s_tail(const string& s) {
        int i = s.find(":");
        return(s.substr(i+1,s.size()));
};

// For searching through to the correct point in the input file
string search_input(const string& flag) {
        string data;
        bool found = false;
        for (int i=0; i<input_string.size(); i++) {
                if (s_top(input_string[i])==flag) {
                        found = true;
                        data = s_tail(input_string[i]);
                        break;
                }
        };

        if (found==false) cerr << "WARNING from search_input --> Required flag
\"" << flag << "\" not found!\n";

        return data;
};
```

E Command and Process Files

```
#!/bin/bash

../source/spinice<<EOF
1.0           #nearest neighbour interaction j
1.0            #mu
0.0 0.0 2.0    #parallel (z-axis) field
5              #number or rows/cols, must be odd
1              #vary the temp, 1=yes, 0=no
0              #vary the field strength
0.0           #starting temp
0.02          #starting field strength
0.766 0.6428 0.0    #field direction (x,y,z), 0,1,0 is along [-1-12] for K trans
0.0433727        #finishing temp
0.001         #large (normal) temperature increment, remember to use -ve if tempstart>tempstop
1.065          #low bound for small increment temperature moves
1.09          #high bound for small increment temperature moves
0.000001         #small increment for temperature, remember to use -ve if temperature is decreasing
0              #equilibrium run switch
3              #lattice initialisation, 1=random, 2='q=x', 3='q=0', 4=r3 x r3, 5=start from file, 6=ferromag. 7=all out 8=random with 0's around the edge (eqv. no PBC) 9=only spin 1's (triangular lattice)
1             #number of measurements at each temp
1              #number of iterations between measurements
0          #number of equilibration steps
2              #monte carlo switch, 1=single MC, 2=loop MC, 3=both
1              #long loops switch, 1=only long loops, 0=long and short loops
f              #perform s(q) neutron mapping. do not leave this true, if the min temp is above this it will create an infinite loop
0.104          #take neutron scattering measurements when this temp is within a temp increment, must be > finishing temp!
500000          #number of loop MCS between each neutron scattering snapshot
50             #number of snapshots to take for a neutron measurement, for ssf's this is the number of times to restart the simulation, for loops this is the number of snapshots to take at the temperature, each separated by the number of MCS specified above
EOF

#NOTES
#this is the control file for the temperature based simulations
# !!!! DON'T FORGET THE SIGN OF THE TEMPERATURE INCREMENTS !!!!
# If the temperature is going up then it will stop one increment before the tempstop value due to the fortran -0/=0 so go one increment further than required.
```

```
#!/bin/bash
#this file processes numerical data from simulations into image files
 etc.

#take the numerical data file and produce graphs of the data
FILE1='?_20*data.out'
FILE2='looplengths*.out'
if [ -a t_20* ]; then
./plottemp.sh $FILE1 $FILE2
echo 'Temperature was varied during the simulation'
elif [ -a f_20* ]; then
./plotfield.sh $FILE1 $FILE2
echo 'Field was varied during the simulation'
elif [ -a eqm_20* ]; then
./ploteq.sh $FILE1
echo 'This was an equilibrium run'
fi

#produce spin maps for each time the showlat or avlat subroutines wer
e called
for i in lattice*.out ; do
./lattplot.sh $i
done

#rename all image files produced by the above operations with appropr
iate extensions
rename 's/\.lattice.png|\.spins.png/.png/g' *.png

#put everything into a folder
mkdir label_me
mkdir label_me/images
mkdir label_me/images/pdf
mv *.out lattpos meas_trans progress row1.spins label_me
mv *.png label_me/images
mv *.pdf label_me/images/pdf
```

```
#script to generate graphs of data
#univ is a switch to plot thermo quantities vs various properties, RE
MEMBER to change the xlabel
#1 = temperature
#12 = temperature/inplane field magnitude
#13 = temperature/t kast
#14 = temperature-t kast

#colours for the pslatex terminal
#0 - dashed black
#1 - red
#2 - green
#3 - dark blue
#4 - magenta
#5 - cyan
#6 - golden yellow
#7 - black
#8 - brown, sim to 1
#9 - lilac
univ=13
gnuplot<<EOF
set terminal pslatex color solid auxfile lw 2
set format xy "$%g$"
#set xlabel 'Temperature'
set xlabel '$ \frac{T}{T_k} $'
# - ENERGY -
set output 't_energy.tex'
set ylabel 'Energy'
set title 'Energy'
set key left
plot '$1' u $univ:18 w lines lt 7 t 'Theory', '$1' u $univ:2 w lines l
t 1 t 'Simulation'
# - MAGNETISATION -
set output 't_mag.tex'
set title 'Magnetisation'
set ylabel 'Magnitude'
set key right
plot '$1' u $univ:16 w lines lt 5 t 'Theory x-comp.', '$1' u $univ:17 w
lines lt 6 t 'Theory y-comp.','$1' u $univ:10 w lines lt 7 t 'Theory Magnit
ude','$1' u $univ:8 w lines lt 1 t 'Sim. x-comp.','$1' u $univ:9 w lines
 lt 2 t 'Sim. y-comp.','$1' u $univ:11 w lines lt 3 t 'Sim. Magnitude'
set output 't_momn.tex'
set title 'Total Magnetic Moment'
set ylabel 'Angle (degrees)'
set y2label 'Magnitude'
set ytics nomirror
set y2tics
plot '$1' u $univ:24 w lines t 'Angle of Moment relative to $ [111] $', '$1' u $u
niv:23 w lines axis x1y2 t 'Magnitude of Moment'
unset y2tics
unset y2label
set ytics mirror
# - ACCEPTANCE PERCENTAGES
set output 't_acceptance.tex'
set title 'Acceptance (single spin flip)'
set ylabel 'Fraction of spin flips accepted'
plot '$1' u $univ:4 w lines t ''
set output 't_loopaccept.tex'
set title 'Loop acceptance'
set ylabel 'Fraction of loops flipped'
plot '$1' u $univ:5 w lines t ''
# - SECONDARY QUANTITIES -
set output 't_spheat.tex'
set title 'Specific Heat'
set ylabel 'Specific Heat'
```

```
plot '$1' u $univ:6 w lines lt 1 t 'Simulation' #'$1' u $univ:19 w lines lt 7 t 'Theory',
set output 't_sus.tex'
set title 'Susceptibility'
set ylabel 'Susceptibility'
#set yrange [0:100]
plot '$1' u $univ:7 w lines lt 1 t 'Simulation'#, '$1' u $univ:15 w lines lt 7 t 'Theory'
set auto
set output 't_magangle.tex'
set title 'Angle of Magnetisation Vector'
set ylabel 'Angle from $ [\bar{1},\bar{1},2] $ direction magnetisation magnitude'
set key right
plot '$1' u $univ:21 w lines lt 7 t 'Theory', '$1' u $univ:20 w lines lt 1 t 'Simulation', '$1' u $univ:8 w lines lt 2 t '$ x $ component of magnetisation'
set key right
set ylabel 'Probability'
set title 'Vertex probabilities'
set output 't_probabilities.tex'
set yrange [-0.1:1.1]
set arrow from 1,0 to 1,1 nohead ls -1
set arrow from 0,0 to 2,0 nohead ls -1
set arrow from 0,1 to 2,1 nohead ls -1
plot 'probabilitiesdata.out' u 1:7 w linespoints t 'pL','probabilitiesdata.out' u 1:8 w lines t 'pR','probabilitiesdata.out' u 1:9 w lines t 'p0','probabilitiesdata.out' u 1:10 w lines t 'pLR','probabilitiesdata.out' u 1:11 w lines t 'pLL'
set noarrow 1
set noarrow 2
set noarrow 3
set output 't_boltzmann.tex'
set yrange [-0.1:2.5]
set ylabel 'Boltzmann weight'
set title 'Boltzmann weight ratios'
plot 'probabilitiesdata.out' u 1:2 w lines t 'w0', 'probabilitiesdata.out' u 1:3 w linespoints t 'wL', 'probabilitiesdata.out' u 1:4 w lines t 'wR'
set auto
set output 't_looplengths.tex'
set title 'Cumulative frequency of looplengths'
set mxtics 10
set xlabel '$ \frac{T}{T_k} $'
set ylabel 'Length of loop'
set zlabel 'Frequency'
splot '$2' w lines t ''
unset output
set term x11
EOF

latex ./tex_for_graphs/showt_energy.tex
latex ./tex_for_graphs/showt_mag.tex
latex ./tex_for_graphs/showt_acceptance.tex
latex ./tex_for_graphs/showt_loopaccept.tex
latex ./tex_for_graphs/showt_spheat.tex
latex ./tex_for_graphs/showt_sus.tex
latex ./tex_for_graphs/showt_looplengths.tex
latex ./tex_for_graphs/showt_probabilities.tex
latex ./tex_for_graphs/showt_boltzmann.tex
latex ./tex_for_graphs/showt_momn.tex
latex ./tex_for_graphs/showt_magangle.tex
dvipdfm showt_energy.dvi
dvipdfm showt_mag.dvi
dvipdfm showt_acceptance.dvi
dvipdfm showt_loopaccept.dvi
dvipdfm showt_spheat.dvi
dvipdfm showt_sus.dvi
```

```
dvipdfm showt_looplengths.dvi
dvipdfm showt_probabilities.dvi
dvipdfm showt_boltzmann.dvi
dvipdfm showt_momn.dvi
dvipdfm showt_magangle.dvi
pdftoppm -png showt_energy.pdf t_energy
pdftoppm -png showt_mag.pdf t_mag
pdftoppm -png showt_acceptance.pdf t_acceptance
pdftoppm -png showt_loopaccept.pdf t_loopaccept
pdftoppm -png showt_spheat.pdf t_spheat
pdftoppm -png showt_sus.pdf t_sus
pdftoppm -png showt_looplengths.pdf t_looplengths
pdftoppm -png showt_probabilities.pdf t_probabilities
pdftoppm -png showt_boltzmann.pdf t_boltzmann
pdftoppm -png showt_momn.pdf t_momn
pdftoppm -png showt_magangle.pdf t_magangle

rm  *.aux *.log t_*.tex *.ps *.dvi
```

```
#!/bin/bash

#script to plot the lattice coordinate file produced by calling showl
at in the sislab program into a human viewable lattice map
gnuplot<<EOF
set border 0
set format y ""
set notics
set nokey
set title '$1'
#set terminal pslatex color solid auxfile lw 1
set format xy "$%g$"
set output '$1.png'
set terminal png size 1000,1000
set size square
plot '$1' w vectors head filled size 0.175,30 lt -1
unset output
set terminal wxt
EOF

#latex ./tex_for_graphs/showt_latt.tex
#dvipdfm showt_latt.dvi
#pdftoppm -png showt_latt.pdf t_latt
#rm *.aux *.log $1.tex *.dvi *.ps
```

```
%file to process the output files from the pslatex terminal of gnuplot into pdf files.
%other thermo quantities are done with a different input file, see plottemp.sh
\documentclass{article}

\special{papersize=395pt,270pt}
\setlength{\paperwidth}{395pt}
\setlength{\paperheight}{270pt}
\setlength{\textwidth}{395pt}
\setlength{\textheight}{270pt}

\topskip0pt
\setlength{\headheight}{0pt}
\setlength{\headsep}{0pt}
\setlength{\topmargin}{-60pt}
\setlength{\oddsidemargin}{0pt}

\usepackage{graphicx}
\usepackage{color}

\begin{document}
 \hspace{-80pt}
 \input{t_mag.tex}
\end{document}
```

```
#!/bin/bash
#control file for the neutron scattering program
../source/neut<<EOF
1                   #lineswitch − 1=do a hori/vert linescan with many points, 0=scan over the q plane
3.0                 #x q multiplier. If this is 2 the extent of the q map is 4pi x b1 by 4pi x b2 which is −2pi to 2pi on each side
3.0                 #y q multiplier. For maps 2 is reasonable, for linescans maybe 3 or 4. For lines only yqmult controls the extent.
0.0                 #qmin. The minimum radius of the qmap for scattering over the plane in the same units as the −hh0 axis. If linescan this is irrelevant.
0                   #hswitch    − 1=do a linescan at constant −hh0 (vertical), 0=constant −k−k2k (horizontal)
0.333                #h_or_k     − if hswitch=1 this is the value of h, if hswitch=0 this is the value of k
EOF

#Linescans are multiplied so that they number of points along a line in q is 10x more than on a map to provide smoother lines.
#A map will be specified as the biggest circle that fits in the rhombus determined by the x and y q multiplier and can also have the center removed using qmin, if any qvector has a magnitude less than q min it will be skipped, then the qmap is a ring not a circle.
```

```
#!/bin/bash
#script for plotting the circular S(Q) maps

#generate a tex file and a ps of the image
gnuplot<<EOF
set terminal pslatex color solid auxfile lw 2
unset key
set output 'sq_unpol.tex'
#set size 20cm,20cm
set format xy "$%g$"
set cbtics axis offset -1 0.0,1.0,3.0
set cblabel offset 2 'Intensity (arbitrary units)'
set palette defined (0 'medium-blue', 3 'skyblue', 6 'yellow', 10 're
d')
set title '{Simulated unpolarised neutron scattering}'
set xlabel '$\left( \bar{h},h,0\right)$'
set ylabel offset 21,0 '$\left( \bar{k},\bar{k},2k\right)$'
set xtics ("-2" -12.56, "-1" -6.28, "0" 0, "1" 6.28, "2" 12.56)
set ytics ('{\small -$\frac{4}{3}$}' -14.51, '-1' -10.88, '{\small -$
\frac{2}{3}$}' -7.26, '0' 0, '{\small $\frac{2}{3}$}' 7.26, '1' 10.88
, '{\small $\frac{4}{3}$}' 14.51)
set size ratio -1
set pm3d map
set cbrange [0:3]
#load 'cir_bz.sh'
splot '$1' u 1:2:3
# set output 'sq_ypol.tex'
# set title 'Simulated in-plane neutron scattering'
# splot '$1' u 1:2:4
# load "cir_bz.sh"
# set output 'sq_pseudo.tex'
# set title 'Simulated pseudospin neutron scattering'
# splot '$1' u 1:2:5
# load "cir_bz.sh"
# set output 'sq_par.tex'
# set title 'Simulated parallel neutron scattering'
# splot '$1' u 1:2:5
# load "cir_bz.sh"
unset output
set term wxt
EOF

#process the tex output into a dvi, then pdf, then png
#tex files are similar to that for processing the magnetisation
latex unpolarised.tex
#latex inplane.tex
#latex pseudo.tex
# latex parallel.tex

dvipdfm unpolarised.dvi
#dvipdfm inplane.dvi
#dvipdfm pseudo.dvi
# dvipdfm parallel.dvi

pdftoppm -png unpolarised.pdf unpolarised
#pdftoppm -png inplane.pdf inplane
#pdftoppm -png pseudo.pdf pseudo
# pdftoppm -png parallel.pdf parallel

#clean up
#rm  *.aux *.log sq_*.tex *.ps
```

```
#draw the BZ boundaries of the first and second pyrochlore and kagome zones onto a pm3d map with size ratio -1, y-axis (-1-12), x-axis (-1 10)

#truncated version for use with circular maps

#variables of useful lengths along the x and y axes when they extend to +-4pi
#1/2*r3 along the y axis, 2* this length is one edge of a pyrochlore BZ
pya=3.627
#halfway along the x axis
pyb=6.283
#2/3 of pyb, the edge of a kagome BZ
pyc=4.189
#
#1st Pyrochlore BZ
# set arrow from pyb,-pya to pyb,pya front nohead ls 4
set arrow from pyb,-pya to pyb,0 front nohead ls 4
# set arrow from -pyb,-pya to -pyb,pya front nohead ls 4
set arrow from -pyb,-pya to -pyb,0 front nohead ls 4
# set arrow from pyb,pya to 0,2*pya front nohead ls 4
set arrow from pyb,-pya to 0,-2*pya front nohead ls 4
# set arrow from 0,2*pya to -pyb,pya front nohead ls 4
set arrow from 0,-2*pya to -pyb,-pya front nohead ls 4
#
#2nd Pyrochlore BZs
#up, right
# set arrow from 0,2*pya to 0,4*pya front nohead ls 4
# set arrow from 0,4*pya to pyb,5*pya front nohead ls 4
# set arrow from pyb,5*pya to 2*pyb,4*pya front nohead ls 4
# set arrow from 2*pyb,4*pya to 2*pyb,2*pya front nohead ls 4
# set arrow from 2*pyb,2*pya to pyb,pya front nohead ls 4
#down, right
set arrow from pyb,-pya to 2*pyb,-2*pya front nohead ls 4
# set arrow from 2*pyb,-2*pya to 2*pyb,-4*pya front nohead ls 4
#set arrow from 2*pyb,-4*pya to pyb,-5*pya front nohead ls 4
#set arrow from pyb,-5*pya to 0,-4*pya front nohead ls 4
set arrow from 0,-3*pya to 0,-2*pya front nohead ls 4
#down, left
#set arrow from 0,-4*pya to -pyb,-5*pya front nohead ls 4
#set arrow from -pyb,-5*pya to -2*pyb,-4*pya front nohead ls 4
# set arrow from -2*pyb,-4*pya to -2*pyb,-2*pya front nohead ls 4
set arrow from -2*pyb,-2*pya to -pyb,-pya front nohead ls 4
#up left
# set arrow from -pyb,pya to -2*pyb,2*pya front nohead ls 4
# set arrow from -2*pyb,2*pya to -2*pyb,4*pya front nohead ls 4
# set arrow from -2*pyb,4*pya to -pyb,5*pya front nohead ls 4
# set arrow from -pyb,5*pya to 0,4*pya front nohead ls 4
#
#1st Kagome BZ
# set arrow from 0.5*pyc,pya to pyc,0 front nohead ls 2
set arrow from pyc,0 to 0.5*pyc,-pya front nohead ls 2
set arrow from 0.5*pyc,-pya to -0.5*pyc,-pya front nohead ls 2
set arrow from -0.5*pyc,-pya to -pyc,0 front nohead ls 2
# set arrow from -pyc,0 to -0.5*pyc,pya front nohead ls 2
# set arrow from -0.5*pyc,pya to 0.5*pyc,pya front nohead ls 2
#
#2nd Kagome BZ's
#up
# set arrow from -0.5*pyc,pya to -pyc,2*pya front nohead ls 2
# set arrow from -pyc,2*pya to -0.5*pyc,3*pya front nohead ls 2
# set arrow from -0.5*pyc,3*pya to 0.5*pyc,3*pya front nohead ls 2
# set arrow from 0.5*pyc,3*pya to pyc,2*pya front nohead ls 2
# set arrow from pyc,2*pya to 0.5*pyc,pya front nohead ls 2
```

```
#right up
# set arrow from pyc,2*pya to 2*pyc,2*pya front nohead ls 2
# set arrow from 2*pyc,2*pya to 2.5*pyc,pya front nohead ls 2
# set arrow from 2.5*pyc,pya to 2*pyc,0 front nohead ls 2
set arrow from 2*pyc,0 to pyc,0 front nohead ls 2
#right down
set arrow from 2*pyc,0 to 2.5*pyc,−pya front nohead ls 2
set arrow from 2.5*pyc,−pya to 2*pyc,−2*pya front nohead ls 2
set arrow from 2*pyc,−2*pya to pyc,−2*pya front nohead ls 2
set arrow from pyc,−2*pya to 0.5*pyc,−pya front nohead ls 2
#down
set arrow from pyc,−2*pya to 0.5*pyc,−3*pya front nohead ls 2
set arrow from 0.5*pyc,−3*pya to −0.5*pyc,−3*pya front nohead ls 2
set arrow from −0.5*pyc,−3*pya to −pyc,−2*pya front nohead ls 2
set arrow from −pyc,−2*pya to −0.5*pyc,−pya front nohead ls 2
#left down
set arrow from −pyc,−2*pya to −2*pyc,−2*pya front nohead ls 2
set arrow from −2*pyc,−2*pya to −2.5*pyc,−pya front nohead ls 2
set arrow from −2.5*pyc,−pya to −2*pyc,0 front nohead ls 2
set arrow from −2*pyc,0 to −pyc,0 front nohead ls 2
#left up
# set arrow from −2*pyc,0 to −2.5*pyc,pya front nohead ls 2
# set arrow from −2.5*pyc,pya to −2*pyc,2*pya front nohead ls 2
# set arrow from −2*pyc,2*pya to −pyc,2*pya front nohead ls 2
#
#3rd Kagome BZ's
#up
# set arrow from −0.5*pyc,3*pya to −pyc,4*pya front nohead ls 2
# set arrow from −pyc,4*pya to −0.5*pyc,5*pya front nohead ls 2
# set arrow from −0.5*pyc,5*pya to 0.5*pyc,5*pya front nohead ls 2
# set arrow from 0.5*pyc,5*pya to pyc,4*pya front nohead ls 2
# set arrow from pyc,4*pya to 0.5*pyc,3*pya front nohead ls 2
#up right
# set arrow from pyc,4*pya to 2*pyc,4*pya front nohead ls 2
# set arrow from 2*pyc,4*pya to 2.5*pyc,3*pya front nohead ls 2
# set arrow from 2.5*pyc,3*pya to 2*pyc,2*pya front nohead ls 2
#right up
#set arrow from 2.5*pyc,3*pya to 3.5*pyc,3*pya front nohead ls 2
#set arrow from 3.5*pyc,3*pya to 4*pyc,2*pya front nohead ls 2
#set arrow from 4*pyc,2*pya to 3.5*pyc,pya front nohead ls 2
#set arrow from 3.5*pyc,pya to 2.5*pyc,pya front nohead ls 2
#right
#set arrow from 3.5*pyc,pya to 4*pyc,0 front nohead ls 2
#set arrow from 4*pyc,0 to 3.5*pyc,−pya front nohead ls 2
#set arrow from 3.5*pyc,−pya to 2.5*pyc,−pya front nohead ls 2
#right down
#set arrow from 3.5*pyc,−pya to 4*pyc,−2*pya front nohead ls 2
#set arrow from 4*pyc,−2*pya to 3.5*pyc,−3*pya front nohead ls 2
#set arrow from 3.5*pyc,−3*pya to 2.5*pyc,−3*pya front nohead ls 2
#set arrow from 2.5*pyc,−3*pya to 2*pyc,−2*pya front nohead ls 2
#down right
#set arrow from 2.5*pyc,−3*pya to 2*pyc,−4*pya front nohead ls 2
#set arrow from 2*pyc,−4*pya to pyc,−4*pya front nohead ls 2
#set arrow from pyc,−4*pya to 0.5*pyc,−3*pya front nohead ls 2
#down
#set arrow from pyc,−4*pya to 0.5*pyc,−5*pya front nohead ls 2
#set arrow from 0.5*pyc,−5*pya to −0.5*pyc,−5*pya front nohead ls 2
#set arrow from −0.5*pyc,−5*pya to −pyc,−4*pya front nohead ls 2
#set arrow from −pyc,−4*pya to −0.5*pyc,−3*pya front nohead ls 2
#down left
#set arrow from −pyc,−4*pya to −2*pyc,−4*pya front nohead ls 2
#set arrow from −2*pyc,−4*pya to −2.5*pyc,−3*pya front nohead ls 2
#set arrow from −2.5*pyc,−3*pya to −2*pyc,−2*pya front nohead ls 2
#left down
#set arrow from −2.5*pyc,−3*pya to −3.5*pyc,−3*pya front nohead ls 2
```

```
#set arrow from −3.5*pyc,−3*pya to −4*pyc,−2*pya front nohead ls 2
#set arrow from −4*pyc,−2*pya to −3.5*pyc,−pya front nohead ls 2
#set arrow from −3.5*pyc,−pya to −2.5*pyc,−pya front nohead ls 2
#left
#set arrow from −3.5*pyc,−pya to −4*pyc,0 front nohead ls 2
#set arrow from −4*pyc,0 to −3.5*pyc,pya front nohead ls 2
#set arrow from −3.5*pyc,pya to −2.5*pyc,pya front nohead ls 2
#left up
#set arrow from −3.5*pyc,pya to −4*pyc,2*pya front nohead ls 2
#set arrow from −4*pyc,2*pya to −3.5*pyc,3*pya front nohead ls 2
#set arrow from −3.5*pyc,3*pya to −2.5*pyc,3*pya front nohead ls 2
# set arrow from −2.5*pyc,3*pya to −2*pyc,2*pya front nohead ls 2
#up left
# set arrow from −2.5*pyc,3*pya to −2*pyc,4*pya front nohead ls 2
# set arrow from −2*pyc,4*pya to −pyc,4*pya front nohead ls 2
```

```
#User inputs for the square ice program. The format is, [hash] label [colon] value. Comments may be placed below.
# RELEASE FILE

# TEMPERATURE PARAMETERS

#Temperature at start: 229

#Temperature at finish: 49

#Temperature increment: 5
# will automatically become −ve if reqd


# FIELD PARAMETERS

#Maximum field: 0.0

#Minimum field: 0.0

#Field increment: 0.1
# will automatically become −ve if reqd

#Field vector: 0.0 0.0
# two numbers representing the x and y components

#Sat field magnitude: 0.0
#the magnitude of the saturation field

#Sat field direction: 0.0 0.0
#the direction of the saturation field


# MONTE CARLO PARAMETERS

#Equilibrium mcs: 2000000

#Saturation mcs: 1
#how many MC steps to take in the saturation protocol

#Mcs per field increment: 2000
# the simulation will do this many mcs at each field value at each temp

#Interaction type: exchange
#can be fourvertex, sixvertex, exchange

# OTHER PARAMETERS

#Number of spins in the lattice: 5000
# must be 2x n^2. examples 18 200 1800 5000 20000 80000

#Starting configuration: random
# can be random, ordered11, randomvertical, dipolegs
```